DISSECTION TECHNIQUE FOR EFFICIENT JOIN OPERATION ON SEMI-STRUCTURED DOCUMENT STREAM

Dong Hyeok Seo, Dong Gyu Lee, Keun Ho Ryu

Chungbuk National University,
hanhwaco@ kornu.ac.kr, dglee@dblab.chungbuk.ac.kr, khryu@dblab.chungbuk.ac.kr

ABSTRACT There has been much interest in stream query processing. Various index techniques and advanced join techniques have been proposed to efficiently process data stream queries. Previous proposals support rapid and advanced response to the data stream queries. However, the amount of data stream is increasing and the data stream query processing needs more speedup than before. In this paper, we proposed novel query processing techniques for large number of incoming documents stream. We proposed Dissection Technique for efficient query processing in the data stream environment. We focused on the dissection technique in join query processing. Our technique shows efficient operation performance comparing with the other proposal in the data stream. Proposed technique is applied to the sensor network system and XML database.

KEY WORDS: Data Stream, Query Processing, Multiple Join, Semi-structured Document

1. INTRODUCTION

There has been increasing interest in the data stream processing. Since the advent of Wireless Sensor Network(WSN) and Ubiquitous Sensor Network(USN), the importance of data stream processing technique have been increased. One of the useful methods to transfer information is to apply semi-structured documents. And the study on the inter-documents query has been vigorously researched.

In recent years, much scholarly work has been done on the topic of join query processing over multiple streams. The reason is increasing application of WSN and exploit of internet. They have enormous data stream that never experienced ever before. Thus, the data stream processing technique for ideal WSN or USN should process rapidly and support fast response to the query processing over the continuously incoming streams. Previous join query processing technique consumes much resource and time. The spot of join query processing is the main memory for rapid response. But it causes physical space constraints in the processing. So, it is necessary to find out the efficient and brief way to process inter-documents join.

In this paper we propose dissection technique for efficient join query processing. Proposed technique dissects the join operation into *estimate the validity*, *pattern match* and *aggregation* as figure 1. At first, we estimate the validity of each semi-structured documents before join processing. The join operation will not process with the not passed document, or process with the passed one. The documents not passed will eliminate without join processing. This estimation can get rid of the meaningless processing and promote the efficient processing. Then we examine the pattern of documents for inter-documents join processing with the passed. This examination will make less effort to join processing. Previous join operation, we have to find out all documents' tree pattern individually.

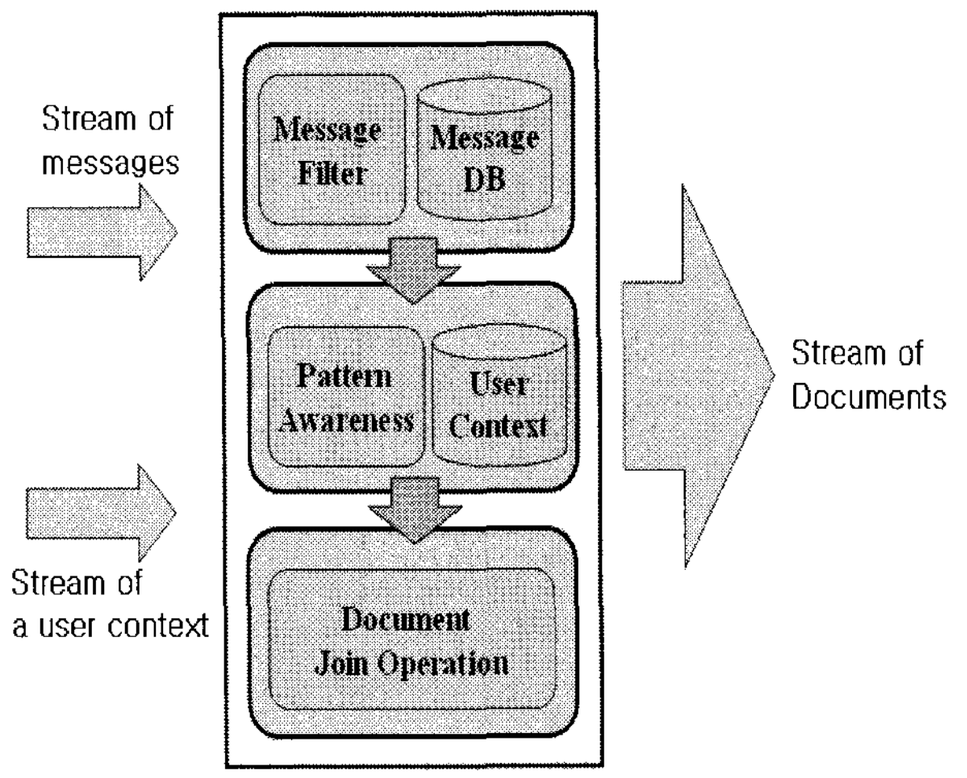

Figure 1. Architecture for efficient join operation

But this examination will support preparation of the rapid real join processing. Thus the join of the multiple documents could reach efficient processing.

2. JOIN QUERY DISSECTION TECHNIQE

We assume join query processing as the dissection object. We dissect the join query processing into 3 parts as mentioned above. They estimate the validity, pattern match and aggregation. The estimation is referred to as filtering. There are various ways filtering semi-structured documents. What is the best way to estimate the documents is not the target of our study. In the same manner, the pattern match technique and aggregation technique themselves are not the main stream of our study in this paper. We propose the dissection technique for efficient join query processing. We dissect join query into 3 parts for the purpose of improving query processing in the data stream. This is beneficial to process the operation within the incoming documents.

With this reason, we estimate the validity of each incoming semi-structured documents at first. We can *estimation* referred to as *filtering* and the techniques of filtering are widely developed. The leading techniques of filtering semi-structured documents are *XFilter* and *YFilter*. A detailed discussion on how to filter documents is beyond the scope of this paper. We would just estimate the validity of incoming documents rather than filtering with complex technology. Second, we examine the tree pattern of documents for join processing. Similarly, there has been much interest in the technique of join query processing on data stream. Structural join has been much interested. The result of this step is essential to join processing in next step. Third, we aggregate the documents that processed by former two steps. We process join operation with former sub-tree. We explain our proposal with example.

Table 1. Expression of data stream

S1	<Event> <Title> <Location> l1 </Location> <Time> t1</Time> </Title> <Target> <Work> w1</Work> <Age> a1</Age> <Sex> s1</Sex> </Target> </Event>
S2	<Visitor> <Profile> <Name> n2</Name> <Age> a2</Age> <Sex>s2</Sex> <Work>w2</Work> </Profile> <Current> <Activity>ac2</Activity> <Location>l2</Location> </Current> <Future> <Activity>ac3</Activity> <Location>l3</Location> <Time>t3</Time> </Future> </Visitor>

Table 1 shows two kinds of documents stream S1, S2. S1 is the message of event. S2 is the profile of visitors. S1, S2 in the example are incoming semi-structured documents form of data stream. The tree pattern of the stream S1 illustrated as Figure 2, and S2 shown in Figure 3.

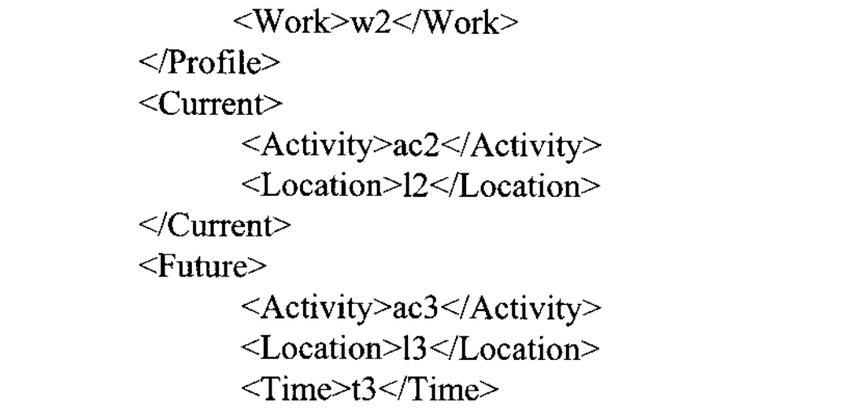

Figure 2. Tree pattern of S1

S1 in Figure 2 is message stream that should deliver event *Topic, Location, Time* according to *Work, Age, Sex* for information broadcasting. And this message stream can input uninterruptedly, iterate, and tremendous stream.

The tree pattern of *Customer* who will get the message illustrates in Figure 3. The information of *Customer* is expressed as *Profile, Current, Future*. Join query process correspond to *Profile* and *Current* should examined whether participate in the *Event* correspond with current *Location, Time, Activity*.

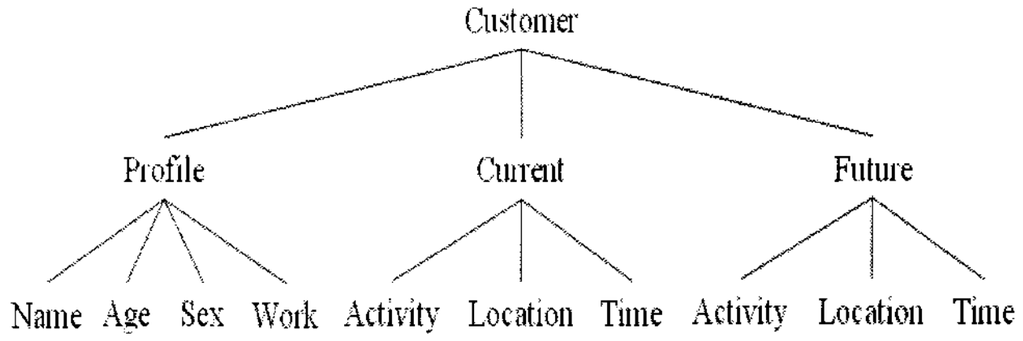

Figure 3. Tree pattern of S2

2.1 Message Filter

We estimate the validity of incoming semi-structured documents that form data stream. Though a lot of documents are incoming, some of them have nothing to do with the purpose of join processing. So there is not necessary for processing join query on every incoming document. This step is available to eliminate the meaningless documents and make less processing on the context awareness system. This is significant step that reduce resource consumption. This estimation step is referred to message filter. The estimation process is described as below.

For filtering the multiple messages, we take advantage of two attributes those are spatial attribute and temporal attribute. New message that is prepared for next step will load on the message database and the same message of following should eliminate.

For ease of exposition, we consider stream S1 in the example. Of course it may be other streams in real system. But we assume just simple environment for make ease of our proposal. They deliver a message about event. We assume each message has a timestamp and dispatch locator. S1 is representative of event messages. It is necessary to estimate the time attribute from the timestamp of the message. In the same way we can estimate the space attribute from the dispatch locator of the message. With these attribute, we can justify the document as the one that has valid context. This leads to clarification of the documents validity.

This filtering step is beneficial for improve performance since it can eliminate redundant messages.

2.2 Pattern-Awareness

In this section, we derive the attributes from the passed messages that justified as efficient documents in the former step. In general, join processing starts with examine tree pattern of each document. Existing method

is inefficient work since require time and the resources with unnecessary operation. For this reason, we derive the attributes those are available for abridge the abundant operation[1][2].

These derived attributes can represent the message and be the key to join with other documents or stream[3]. We can use these attributes for match with the context of *Visitor*. It is refer to this match as *pattern match*. We can make use of this *pattern match* for next step – join processing or aggregate two information streams. This is beneficial for next step in the side of economize resources.

In the S1, the node *Target* is remarkable. S1 announce the event and it provides S2 with useful information. S2 is *Visitor*'s profile acknowledged by main entrance tag reader.

Node *Target* of S1 has three attributes *Work, Age, Sex*. As we try to produce advanced information, we can find out the participants of event refer to the value of three attributes. Since S2 has similar attributes under the node *Profile,* they are remarkable too. Instead of make an entire investigation into whole tree pattern of each document of stream S1, we just examine the sub-tree of each document on the basis of the derived attributes. Comparing *Profile* node (in S2) and *Target* node (in S1) results out significant solution. We can make the list of participants of each event or guide the visitors to proper place of event at time. It is important to examine this kind of attribute or pattern that will decrease the number of operation in the data steam. In the sensor network, there are too many messages comes from the sensor. This examination is profitable process for continuous data stream process.

2.3 Document Aggregation

In this section, we aggregate the message document and context of visitor. For efficient process, we make the most of the result of former steps. The representative pattern derived from documents passed the estimation is beneficial for final aggregation[1]. We just process the join query process. Just search comparing the representative node value within two incoming stream documents. And then go on the aggregation process. We can reach more advanced performance of join process within multiple semi-structured documents (e.g. data stream on the wireless sensor network service). This process can afford to rapid performance. Furthermore, we can develop application services with this aggregation results (e.g. intelligent announcement and guide system)

3. PROPOSED ALGORITHM

Given input documents E and U, we eliminate the invalid documents that should not go on processing. By the use of spatiotemporal attribute of each document, we can drop invalid documents, in Line1, Line2 and Lin3 of Algorithm.

Then we examine the sub-tree pattern for rapid join operation of multiple documents, in Line 4, Line5 of Algorithm. We can get rapid join processing over the multiple documents, in Line 6 of Algorithm.

Algorithm Dissection(E, U)
INPUT E: semi-structured documents of event message
U: semi-structured documents of user context
OUTPUT joined semi-structured document
BEGIN
1:Check tree pattern from semi-structured documents
2:If E and U are same values of spatiotemporal attribute in message DB
3:Then remove E and U
4: **for** passed estimation documents **do**
derive the representative sub-tree of E and U
5: **Match** the sub-tree of E into tree pattern of U≠ ∅
6: Perform the join operation as E▷◁ U
END

Figure 4 Dissection Algorithm

4. CONCLUSION AND FUTURE WORK

In this paper, we proposed the dissection technique for efficient join processing in the data stream transfer system. Message filtering, pattern awareness and value aggregation are beneficial for advanced query processing. This proposal will guide reducing the cost of load and efficient join processing. We would like to evaluate this dissection technique in practice. And we would like to adopt various context-awareness systems.

References

[1] M. Hong, A. Demers, J. Gehrke, C. Koch, M. Riedewald,
W. White. Massively Multi-Query Join Processing in Publish/Subscribe Systems. In Proc. SIGMOD, 2007

[2] F.Fabret, H.-A. Jacobsen, F. Llirbat, J. Pereira, K. A. Rosa, and D. Shasha. Filtering algorithms and implementation for very fast publish/subscribe. In Proc. SIGMOD, pages 115-126, 2001.

[3] J. Bae, B. Moon, S. Lee, Segment Join Technique for Processing XML Queries Fast. The Korea Institute of Information Scientists and Engineers, pages 55-58, 2005.

ACKNOWLEDGEMENT

This research was supported by a grant (#07 국토정보 C05) from Cutting-edge Urban Development – Korean Land Spatialization Research Program funded by Ministry of Construction & Transportation of Korean government.