INTEGRATED DESIGN OF EVENT STREAM SERVICE SYSTEM ARCHITECTURE (ESSSA)

Kyungeun Park, Yanggon Kim
Department of Computer & Information Science, Towson University, 7800 York Road, Towson, MD 21252, USA

Jinkyu Lee*, Juno Chang**
*ReegySys, Inc., 456 Dogok-dong, Gangnam-gu, Seoul, 135-270, Republic of Korea
**Div. of Media Technology, Sangmyung University, 7 Hongji-dong, Jongno-gu, Seoul, 110-743, Republic of Korea

Keywords: Event Stream Management, Event Query Language (EQL), Radio Frequency Identification (RFID), Ubiquitous Sensor Network (USN), Object-Relational Spatial Database Management System.

Abstract: This paper proposes the integrated design of an event stream service system architecture (ESSSA) which can be used as a gateway to capture and process enormous amounts of event streams continuously generated by radio frequency identification (RFID) or variable sensors over ubiquitous sensor network (USN) installed in a wide variety of locations, transfer the refined events to the various event-driven applications connected to this architecture, and allow the corresponding actions to be taken by the applications. The events continuously delivered from versatile origins and their streams are inherently uncertain, unbounded, and time-varying. Their arrival rate might severely fluctuate in some cases. Accordingly, this design focuses on the elaborate handling of the event streams by facilitating robust event gathering scheme as a front-end gateway in combination with a back-end event processing engine. The paper also contains the time-varying event query processing schemes of the ESSSA by extending the object-relational spatial database management system, ZEUS. The ESSSA can be applied as a framework for interfacing RFID or USN sensor based applications such as an enterprise resource planning system (ERP), a supply chain management system (SCM), a warehouse management system (WMS), or a centralized command and control system.

1 INTRODUCTION

Today's highly networked environments encouraged by rapid growth of highly advanced wireless telecommunication and corresponding software technologies encourage us apply them to real world and take fully advantage of the conveniences and efficiencies they present. To do so, it necessarily follows that we have to consider the effective way to handle continuously generated events or signal streams in such ubiquitous computing environments. Events streams are to be captured from the event origins, filtered by the predefined conditions, maintained within the efficient events archive and delivered to the appropriate service applications depending on the event status.

Technical considerations regarding on the related issues are required to design the event stream service system architecture (ESSSA) that can be used as an infrastructure for handling enormous amounts of event streams generated by all sorts of event origins which are equipped with radio frequency identification (RFID), sensors of ubiquitous sensor network (USN), or conventional wireless telecommunication facilities. In this paper, the event capturing system built on top of the main memory is to support real time transactions and enables inflow of the enormous events to be captured and evaluated instantaneously. Disk based conventional database management system (DBMS) is used as a backup repository for the post-analysis to the once accumulated event history data.

Event data discussed in this paper is inherently uncertain, unbounded, and time-varying and their arrival rate from various event origins might severely fluctuate in some cases. For this reason,

Park K., Kim Y., Lee J. and Chang J. (2007).
INTEGRATED DESIGN OF EVENT STREAM SERVICE SYSTEM ARCHITECTURE (ESSSA).
In *Proceedings of the Second International Conference on e-Business*, pages 51-56
DOI: 10.5220/0002113400510056

elaborate consideration has to be taken into account to capture and maintain the events, evaluate the pre-defined event rules and react according to the corresponding status of the events.

On receiving the events, the Event Collector of the ESSSA pre-analyzes and classifies the categories to which individual event belongs. The recognized events are piled on the associated event queues which are constructed by event stream within the main memory, while the Event Query Manager, important sub-component of the ESSSA parses the event queries requested by the service applications and extracts the event monitoring rules. They are registered to the event monitoring rule tables in order for the continuously incoming events to be evaluated immediately as soon as they are identified as satisfying events to a rule of interest.

Simple event is interrelated between events and may imply different meaning according to the context and the arrival of time. In this paper, we extended the spatial query language supported by ZEUS, object-relational spatial DBMS (Park, et al, 1998), to allow the ESSSA to define events and manipulate their behavior and the relationship beteen them. As the events received from ubiquitous equipments may exist in different spatial locations, using the spatial fearures of ZEUS system makes the target system more powerful and easily applicable to the real world applications.

It is required to provide a standard interface that facilitates seamless connection across add-on service applications. To resolve these requirements, service-oriented architecture (SOA) interface technology is selected as an interface infrastructure. This will provide the plug-and-play interfaces and accelerate the extension of the existing service framework with the new services considered to be attached.

This paper is composed of 5 sections. This section introduces the research background and the fundamental technologies applied to the design of the ESSSA. In section 2, the previous related research and our current work on event stream processing is presented. Section 3 discusses about the main components of the ESSSA in detail. The fourth section describes the issues related to the processing event streams in the ESSSA. Finally, we conclude the paper with a discussion of the proposed system and current and future implementation issues.

2 RELATED RESEARCH AND OUR WORK

2.1 Related Research

In handling events, a lot of research efforts have been made in the last decade to formalize the event, the behaviour of the events, and the relationship between them.

The study on event streams was drived from the concern for handling time-varying data within time-series management system. Then, the active database rose to the surface of event-based temporal reasoning. (Motakis, Zaniolo, 1997). Active database has been frequently compared with data stream management system. Active database uses triggers to support automatic response to events. The action may change the value of the database, whereas, the rapid materialization of ubiquitous computing environments revitalized the events stream handling issue. Data stream management system demands complex events processing schemes to treat inflow of events from the sources and supports the continuous queries over the running streams. (Luckham, 2001), (Rizvi, 2005).

Many researchers have analyzed events and distinguished the nature of the event queries over event streams in time from the traditional queries on the relations. (Motakis, Zaniolo, 1997), (Zimmer, Unland, 1999), (Luckham, 2001), (Arasu, Badu, & Widom, 2003).

The most important difference is that the event initiates the event query, while the ordinary query which is used in the traditional DBMS is self-activated. (Chandrasekaran, et al, 2003). Time-varying event query needs to keep the event history in event repository and sliding window mechanism to constraint the number of event at a particular point of time has been chosen as a soultion. (Ghanem, et al, 2007).

Also, semantic formalization such as basic definitions and classification of events, their time-varying relationships, and the events rules have been accomplished from different perspectives. (Motakis, Zaniolo, 1997), (Zimmer, Unland, 1999), (Arasu, et al, 2003), (Chandrasekaran, et al, 2003), (Rizvi, 2005), (Bry, Eckert, 2006), (Bry, Eckert, & Pătrânjan, 2006).

Many event stream management systems such as TelegraphCQ (Chandrasekaran, et al, 2003), STREAM (The STREAM Group, 2003), and REMS (Hwang, Cheong, Kim, & Lee, 2005) have been developed as the solutions to the challenges discussed so far.

2.2 Design of the Event Stream Service System Architecture (ESSSA)

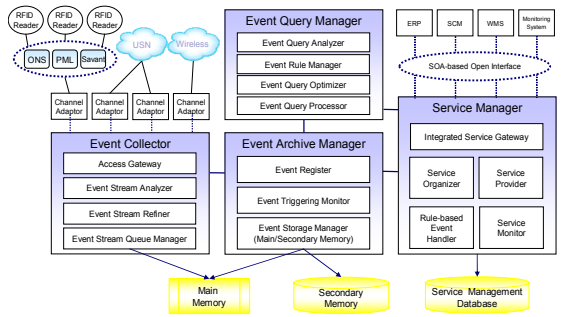

Figure 1: The Event Stream Service System Architecture.

Figure 1 shows four major sub-systems: the Event Collector, the Event Query Manager, the Event Archive Manager, and the Service Manager.

The ESSSA considers fundamental issues described in the previous section and targets to handle event streams from massive RFID tags, sensor data of USN, and any other spatiotemporal information of any moving object.

To satisfy this objective, the ESSSA is to extend the object-relational spatial DBMS, ZEUS (KTDATA, 2005), from the lower storage management level to the higher query processing level.

3 MAIN COMPONENTS

3.1 Event Collector

At the front-end of the Event Collector is the Access Gateway, which receives immense amounts of event streams that have been classified and recognized through the event pre-processors such as object naming service (ONS) server, physical markup language (PML) server, and Savant server as defined in EPCglobal architecture framework standard (EPCglobal, 2005). The events are originated from various RFID readers, sensors in USN, wireless communication network entities.

The Event Collector also includes the Event Stream Analyzer, the Event Stream Refiner and the Event Stream Queue Manager. They categorize events, input them in the event stream queue, and filter out duplications among adjacent readers or meaningless events holding the unchanged value. The events refined in this level are ready to be monitored and evaluated in the Event Query Manager and broadcast to the external service applications such as enterprise resource planning system (ERP), supply chain management system (SCM), warehouse management system (WMS), tracking system, or centralized command and control system.

3.2 Event Query Manager

The Event Query Manager processes event queries, which perform the event rules requested from the service applications and provide them with the proper services in time. The Event Query Analyzer, the Event Rule Manager, the Event Query Optimizer, and the Event Query Processor are composing the Event Query Manager.

The Event Query Analyzer parses event query statements written in the extended structured query language (SQL) query language and extracts the event rules and registers them to the appropriate event monitoring rule tables.

The Event Rule Manager gathers event rules from event monitoring rule tables which are divided into Post-Event Monitoring Rule Table and Pre-Event Monitoring Rule Table.

The Event Query Optimizer makes a plan for the execution of the complex event query. Finally, Event Query Processor processes the event query according to the query plan.

3.3 Event Archive Manager

The Event Archive Manager consists of the Event Register, the Event Triggering Monitor, and the Event Storage Manager. The Event Register stores events coming from the Event Collector to the event repository.

The Event Triggering Monitor synthetically monitors registered events, determines if the events are matched with the predefined triggering rules, and reports them as query results. The associated business service applications are then to be informed of the query results through the Service Manager.

The Event Storage Manager maintains the events by constructing spatiotemporal index on them depending on their (symbolic/physical) locations and time of occurrences, keeps a snapshot of the events data of the main memory repository, and performs swapping between the main memory repository and the secondary backup DBMS.

3.4 Service Manager

The Service Manager contains the Integrated Service Gateway, the Service Organizer, the Rule-based Event Handler, the Service Provider, and the Service Monitor. The Integrated Service Gateway allows external service applications to interact the ESSSA with SOA-based open interfaces. The Service Organizer registers services in combination with the event conditions. The Rule-based Event Handler transforms registered services into event queries and requests the associated query results. The Service Provider provides external business service applications with web services to broadcast the event occurrences and their status in real-time. The Service Monitor controls the enrolled service conditions and then the ESSSA will identify the occurrence of the event and provide the related information through Service Provider.

4 EVENT PROCESSING

This section mainly discusses the key concepts and major extension details of the ESSSA to effectively process the event streams. They are composed of event collecting process, query language extension, and event querying process.

4.1 Event Collecting Process

In order to handle the massive event streams from many different origins, this process acquires, analyzes and refines, and manages event stream queues by phases.

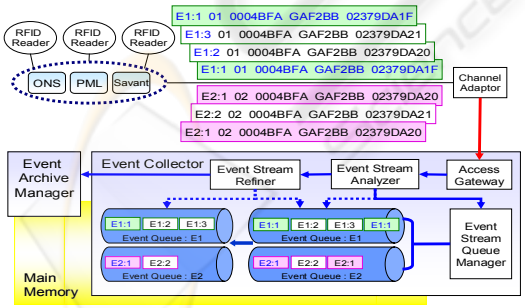

Figure 2: The Event Collecting Process.

As depicted in Figure 2, this architecture assumes that events are transmitted from RFID tags, gathered through ONS, PML and Savant server and sent to the Access Gateway via Channel Adaptor of Event Collector. Initially acquired events are delivered to the Event Stream Analyzer, which breaks down the tag information into separate meaningful values such as the classification and the identification information of the tagged object. The electronic product codes (EPCs) system for RFID tag is used to uniquely identify a single object. Generally, the EPC tag contains a header, the EPC manager's information, object class, manufacturer, product identification information, serial number, and so on. (EPCglobal, 2005). The Event Analyzer classifies the events and inputs them into the separate event stream queues.

The Event Stream Refiner filters out duplications among adjacent readers or removable events, which hold the same value as the one of the previous event. The events refined in this level are ready to be monitored and evaluated based on the Event Query Manager according to the event rules. The query results are then replied to the Service Manager to be delivered to the external add-on service applications.

In figure 2, events starting with "E1:1" are duplicated and the Event Stream Refiner eliminates the last one. In some cases, we should not cut the duplicate to keep track of the status of the object. Thus, an event query language (EQL) has to support capabilities to define these constraints to the rules depending on the specific characteristics of events.

4.2 Query Language Extension

The extension aims to accommodate characteristics of events and represent them. Specifically, event handling operators provides the service applications with fundamental means to express various event rules. This section describes the extension details by presenting simple query statements, which express event related tasks. The base query language for extension is from the spatial query language of object-relation DBMS, called ZEUS. (Park, et al, 1998), (KTDATA, 2005).

The following section describes detailed features of the query language extension for event processing.

4.2.1 Definition of Event Relation

The syntax for event definition is a variant of the relations definition in conventional DBMS.

```
create event PassGate (epcID
       char(100), gateID char(15),
       location point);
create event StoreToShelf (epcID
       char(100), shelfID char(15),
       noBox integer);
create event LiftFromShelf (epcID
       char(100), shelfID char(15),
       noBox integer);
```

```
create event LoadedToTruck (epcID
       char(100), truckID char(15),
       noBox integer);
```

In the above examples, the keyword "event" is specified to alert us that a tuple of an event relation implicitly contains an "event_time" attribute typed as "timestamp" to hold the time of the event as well as the explicitly indicated attributes. Additionally, the system performs the event-related prerequisite tasks such as preparing the event queues and registering it to an anticipating event list.

4.2.2 Relationship Operators between Events

Event relationship operators are classified into three groups: temporal relationship, logical relationship, and aggregation relationship.

Generally, temporal relationship operators distinguish an event's temporal order and act depending on the individual operator. The following shows the way that temporal operators are used in "where" clause of traditional "select" query statement:

```
select p.epcID, p.event_time,
       s.event_time, s.shelfID
from   PassGate p, StoreToShelf s
where  PassGate  happen before
       StoreToShelf within
       [p.event_time,
       p.event_time+10];

select p.epcID, p.event_time,
       l.event_time, l.shelfID
from   PassGate p, LiftFromShelf l
where  PassGate happen after
       LiftFromShelf;
```

The first SQL statement finds the object, which has passed the gate and is stored on the shelf for up to 10 minutes. The second one recognizes the object passing the gate after it has been lifted from the shelf without any time limitation.

Another type of event operator is to simply figure out the logical relationship between events such as conjunction or disjunction. Furthermore, query language supports negation of an event to ensure an event does not happen.

```
select p.epcID, p.event_time,
       s.event_time, l.event_time,
       l.truckID, l.noBox
from   PassGate p, LoadedToTruck l
where  PassGate and LoadedToTruck
        within [t, t+10];
select s.event_time, s.shelfID
from  StoreToShelf s,LiftFromShelf l
where StoreToShelf or not
       LiftFromShelf within
       [t, t+30];
```

Moreover, extended query language provides the aggregate operators, which gather the events of interest and evaluate the operations: count( ), sum( ), and average( ).

```
select count(PassGate),
from   PassGate p
where  p.epcID like '01 0004BFA%';

select sum(l.noBox),
from   LoadedToTruck l
where  LoadedToTruck within
       [t, t+10];
```

The above two event query statements calculate the number of objects having tags, which start with '01 0004BFA' and the number of whole boxes loaded to a truck within a specified time period by using the aggregate operators.

4.3 Event Querying Process

In the Event Query Manager, the Event Query Analyzer begins parsing the event query statements and registers the extracted event rules to the corresponding event monitoring rule tables, which are handled by the Event Rule Manager. The event rules that are supposed to be monitored are thoroughly examined in order to schedule and arrange the events within the event monitoring map table (EMMT).

Table 1: The Event Monitoring Map Table.

Event	Successor	Predecessor	Service ID
E1	E2	E1	S1, S6
E2	E3	E1	S2, S7
E3	E6	E7	S3, S8
E4	E4	E6, E3	S4, S9, S10
E5	E1	-	S5

As one single event can be related with multiple event queries, the EMMT is introduced to synthetically handle the events to be monitored. The event map is associated with three sub-components of the Event Query Manager. First, the Event Rule Manager registers the event and builds up the map, secondly, the Event Query Optimizer periodically investigates the correlations between events, and lastly, the Event Query Processor performs the actual queries.

By coupling the main memory repository scheme and the disk-based DBMS, we intend to take advantage of the benefits provided by both systems. The continuously generated massive events need to be processed in real time. In addition, the time-varying results of the event query should be produced to satisfy the requirements of the various service applications based on the complex event queries.

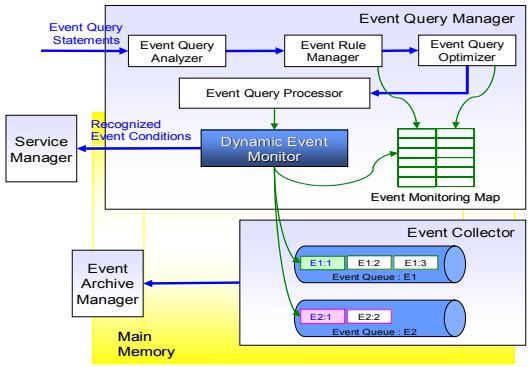

Figure 3: The Event Querying Process.

Figure 3 shows the event querying process in combination with the sub-components of the ESSSA.

5 CONCLUSIONS

This research results from the increasing requirements relevant to the emergence of the innovative sensing technologies related with RFID, USN, and any other wireless telecommunication facilities. Consequently, the elaborate handling of the massive amounts of events are regarded as the key success factor of the event stream service framework, which coordinates various event-driven real world applications.

In this paper, we have designed the ESSSA based on the object-relational spatial database management system, ZEUS. We extended the spatial query language in order to treat the time-varying nature of the event streams. Furthermore, as an event stream is related with multiple queries and enormously increased over time, the efficient event sharing and the flexible switching mechanism needs to be thoroughly considered when designing and implementing the main memory repository.

The event query language suggested in this paper is to be formalized to guarantee the completeness of the functionalities. Moreover, the query optimization procedures of the existing system should be refined to support the event queries in the future work.

REFERENCES

Arasu, A., Badu, S., & Widom, J., 2003. CQL: A Language for continuous queries over streams and relations. In *DBPL*.

Bry, F., Eckert, M., & Pătrânjan, P., 2006. Querying composite events for reactivity on the Web. In *Proc. Intl. Workshop on XML Research and Applications*, number 3842 in LNCS, pages 38-47. Springer.

Bry, F., Eckert, M., 2006. A-High-level query language for events. In *Proc. of the IEEE Services Computing Workshops (SCW'06)*.

Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M. J., Hellerstein, J. M., Hong, W., Madden, S., Raman, V., Reiss, F., & Shah, M., 2003. TelegraohCQ: Continuous dataflow processing for an uncertain world. In *CIDR 2003, First Biennial Conference on Innovative Data Systems Research*, Asilomar, CA, USA.

Ghanem, T. M., Hammad, M. A., Mokbel, M. F., Aref, W. G., & Elmagarmid, A. K., 2007. Incremental Evaluation of Sliding-Window Queries over Data Streams. In *IEEE Trans. on Knowledge and Data Engineering*, vol. 19, no 1.

Hwang, J. G., Cheong, T. S., Kim, Y. I., & Lee, Y. J., 2005. Trends of RFID Middleware Technology and Its Application. In *Telecommunication Trend Analysis and Survey*, vol 20. no 3.

Luckham, D., 2001. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems.* Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA,

Motakis, I., Zaniolo, C., 1997. Temporal aggregation in active database rRules. In *Proc. Inlt. Conf. on Management of Data(SIGMOD).* ACM Press.

Park, K. E., Lee, J. K., Lee, K. J., Ahn, K. H., Lee, J. W., & Kim, J. S., 1998. The development of ZEUS(GEUS): A spatial DBMS tightly integrated with an object-relational database engine. In *URISA 1998 Annual Conference Proceedings*, Charlotte, NC.

Rizvi, S., 2005. Complex event processing beyond active databases: Streams and uncertainties. Technical Report No. UCB/EECS-2005-26.

The STREAM Group, 2003. STREAM: The Stanford stream data manager. In *IEEE Data Engineering Bulletin*. vol. 26, no. 1.

Zimmer, D., Unland R., 1999. On the semantics of complex events in active database management systems. In *Proc. Intl. Conference on Data Engineering (ICDE)*, pages 392-399. IEEE Computer Society Press.

KTDATA, 2005. User Manual of ZEUS DBMS, vol. I, II., http://ktdata.co.kr.

EPCglobal, 2005. The EPCglobal Architecture Framework. From http://www.epcglobalinc.org/standards/Final-epcglobal-arch-20050701.pdf.