Clustering based on Dependence Tree in Massive Data Streams

Hong-won Yun, *Member, KIMICS*

Abstract— RFID systems generate huge amount of data quickly. The data are associated with the locations and the timestamps and the containment relationships. It is requires to assure efficient queries and updates for product tracking and monitoring. We propose a clustering technique for fast query processing. Our study presents the state charts of temporal event flow and proposes the dependence trees with data association and uses them to cluster the linked events. Our experimental evaluation show the power of proposing clustering technique based on dependence tree.

Index Terms—Clustering, RFID Data, Data Association, Dependence tree.

I. INTRODUCTION

RFID (Radio Frequency Identification) technologies offer practical benefits to almost anyone who needs to keep track of physical assets. The benefits of RFID technology are to create a physically linked world where every object is monitored and tracked. RFID applications are set to play an essential role in object tracking and supply chain management systems. In addition to emerging applications in retail and distribution, RFID has been gradually adopted and deployed in a wide area of applications, such as high tolls, supply chain and logistics, healthcare and access control [1-3]. It is expected that every major retailer will use RFID systems to track the movement of products from suppliers to warehouses, store backrooms and eventually to points of sale [4].

RFID holds the promise of real time identifying, locating, tracking and monitoring physical objects. RFID data has to be collected, filtered, and transformed into semantic application data. Since RFID data contains false readings and duplicates, such data have to be filtered and cleansed [5]. In general, RFID data are generated quickly and is in high volume, its detection usually demands efficient processing. Although there are effective methods to filter RFID data, high volume of data can cause of slower queries and updates. Thus, RFID delivers a granularity that needs to be effectively managed from a quantity as well as quality perspective. One of the biggest problems for RFID applications is how to deals with huge amount of generated RFID data.

Such huge amount of data and various queries pose great challenges to relational database since the processing may involve fast retrieval over a large number of inter-related tuples through different stages of object movements. This paper aims to formalize the data association and to propose the clustering way using it. In this paper, we introduce the data association over huge volume of RFID data and propose efficient algorithm for RFID data clustering, including removal not valid data.

The paper is organized as follows. We first introduce the background of RFID data features and clustering in Section II. Then in Section III we propose state charts of event flow, including the dependence trees and discuss algorithms for clustering using data association. Next we show performance evaluation of clustering based on proposed method in Section IV. Finally we conclude our study and discuss the related issues in Section V.

II. BACKGROUND

A. Characteristics of RFID Data

Radio Frequency Identification (RFID) systems consist of three components: RFID transceivers or readers, RFID transponders or tags, and the application that makes use of the generated data [1,6]. The electronic product code (EPC) standard [7] defines world wide unique IDs for RFID tags which are stored in their memory and used to uniquely identify each of them. Despite of diversity of RFID applications, RFID data share common fundamental characteristics [8], which can be listed as follows:

Simple Data: Data generated from an RFID application can be seen as a stream of RFID tuples of the form (EPC, location, time) [4], where *EPC* is an Electronic Product Code which university identifies an item [9]. *Location* is the place where the RFID reader scans the items, and *time* is the time when the reading took place.

Large volume of data: One of the biggest issues in RFID is to deal with the in-flood of data. The volume and velocity of RFID data will exceed the capacity of existing technology infrastructure. As an example, Wal-Mart is expected to generate 7 terabytes of RFID data per day [4]. Even modest RFID deployment will generate gigabytes of data a day since each item is tagged and will send the data continuously about its EPC, location and time [11].

Inaccuracy: One of the primary factors limiting the

Manuscript received March 6, 2008; revised May 15, 2008. Hong-won Yun is with the Department of Information Technology, Silla University, Busan, 617-736, Korea (Tel: +82-51-999-5065, Fax: +82-51-999-5657, Email: hwyun@silla.ac.kr)

widespread adoption of RFID technology is the inaccuracy of the data stream produced by RFID readers. The observed read rate accuracy in real RFID deployment remains still on average in the 60-70% range [10]. We need to clean this data before feeding our system with such unreliable data. As a result, the data in RFID are generally inaccurate.

Spatial and Temporal: RFID observations are dynamically generated and the data carry state changes [12]. All RFID observations are associated with the timestamps, the objects' locations and the containment relationships change along the time [11,15]. It is essential to model all such data by an expressive data model suitable for application level interaction including tracking and monitoring.

In this paper, we address some of the issues in RFID data management and focus on the large volume of data above mentioned.

B. System Architecture

In [13] Mark Palmer suggests seven common principles of the effective RFID data management which will ensure reliable, real time, accurate decisions. The first principle is *Digest the Data Close to the Source*. The second principle is about transformation simple events into more complex meaningful events. The third principle is called *Buffering Event Streams* and *Cashing Context* is the next forth principle. The fifth principle is *Federate Data Distribution in Near Real Time*. The principle *Graceful Aging RFID Data* deals with large volume of generated RFID data. The last principle is *Automate Exception Handling*, it should be considered in mind if you want to achieve the successful RFID data management.

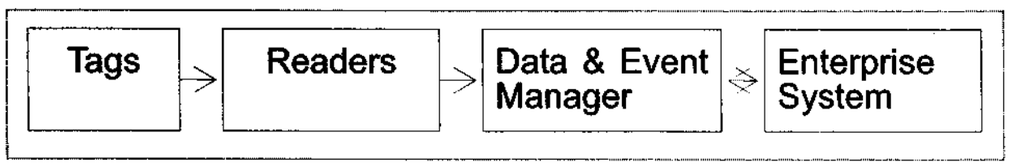


Fig. 1 System Architecture

Fig. 1 suggests a layered architecture for managing RFID data. The lowest layered consists of RFID tags attached to different items and pallets. This layer is responsible for coordinating multiple tagged objects and cleaning data before sending to the next layer. The next is the layer of RFID reader which is deals with the stream of tuples. This layer includes the first principle and the second principle in Mark Palmer's seven principles. The Data & Event Manager level is responsible for mapping the low level data from readers to a more suitable form that is useful at application layer. This layer is generally called middleware, plays the main role in RFID data management and typically deployed between the readers and the application level in order to correct data streams and send cleaning data to enterprise system layer.

One of the primary factors limiting the widespread adoption of RFID technology is the unreliability of the data streams produced by RFID reader [10,14]. Such error rates render raw RFID data streams essentially useless for the purpose of higher level applications. RFID middleware are deployed between the readers and the applications such as enterprise system in this paper. This middleware includes from the third to the fifth principles in Mark Palmer's principles.

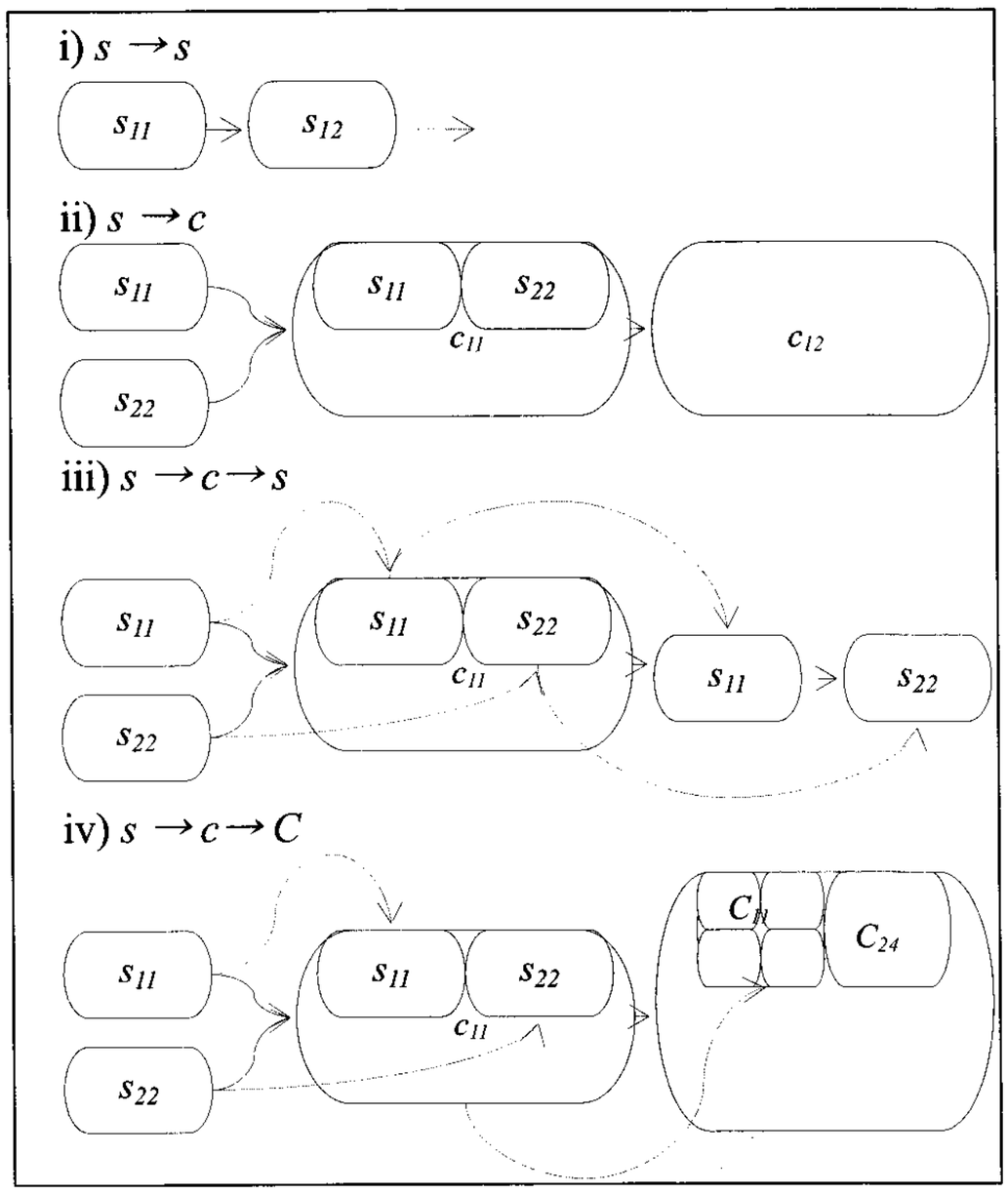
Applications on the enterprise system layer can interact with the middleware layer by issuing queries as well as by installing standing queries that result in a stream of matching data [11]. The last layer deals with huge amount of generated RFID data streams. A manager is involved in enterprise system in order to provide fast response to different queries. In this paper we study state charts and dependence trees assigned with the enterprise system in the architecture and developing clustering technique of the RFID data streams.

III. TEMPORAL EVENTS AND DEPENDENCE TREES WITH DATA ASSOCIATION

A. Temporal Events

In traditional event processing system there could be many types of events such as single events and composite events. Here we suppose that the continuous tag readings are under the model of append only relations for RFID data streams. We consider two types of events that are single events and composite events. Composite events are temporal patterns formed by other events which can be two more single events or composite events. Readers are used to detect automatically the presence of product, package, and packing boxes.

A single event with a single item and composite event with containment relationship can be classified as follows:



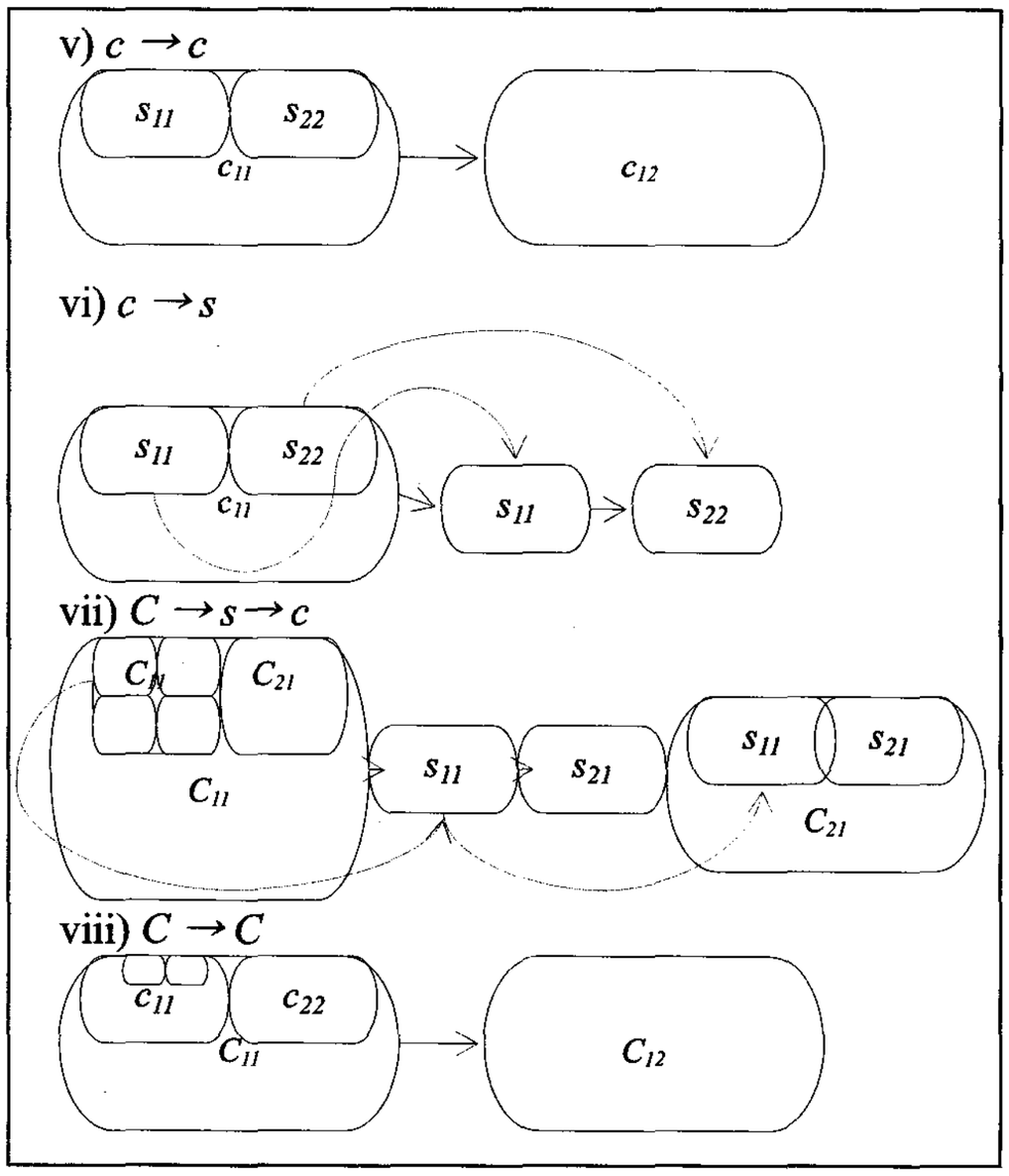


Fig. 2 State charts of temporal event flow

In Fig. 2, $s \rightarrow s$ say that reader *r1* scans a single item and then detect a single temporal event, and next reader *r2* read same single item at another location then detect a different single temporal event. The lowercase *c* in Fig. 2 is contained two or more single events, a single event could be only primitive events we consider. Since there could be many different types of composite events such as packaging boxes, capital *C* in above Fig. 2 means two or more composite events. Also, we start by defining a deterministic single event, which is a named tuple of the basic form *EventType* (*ProductID*, *ReaderID*, *Location*, *Timestamp*). Data generated from an RFID application can be seen as a stream of RFID tuples of the above form as showed related previous work many times.

Here we use three types of events to presents a single event type, composite events and join for both of single and composite events as follows:

s-events = (*Pid, Rid, Lid, tst, tet*)
c-events = (*Pkid, Rid, Lid, tst, tet, amt*)
sc-events = (*Pkid, Pid*)

Fig. 3, Fig4, and Fig5 present the sample events in the RFID database after cleaning. For example, s-table(1, R01, L01, 100, Null) represents 1 is the product id, R01 is Reader id, 100 is transaction start time and Null is transaction end time (*i.e.*, *now*).

s-table

Pid	Rid	Lid	tst	tet
1	R01	L01	100	Null
1	R02	L02	200	300
2	R01	L01	110	Null

Fig. 3 Sample single events

c-table

Pkid	Rid	Lid	tst	tet	amt
1000	R11	L11	400	Null	12
1000	R22	L22	500	600	12
2000	R11	L11	410	Null	36

Fig. 4 Sample composite events

sc-table

Pkid	Pid
1000	1
1000	2
2000	1

Fig. 5 Sample relationship events of single and composite events

B. Dependence trees with data association

We present the dependence trees with data association such as Fig. 6, Fig. 7, and Fig. 8. For example, the event s_{11} associates next events c_{11} and s_{11} in the dependence tree with data association (*s,c,s*) shown in Fig. 6. After generating the dependence tree with data association, we can use them to cluster the associated events in which to execute rules.

a) $s \rightarrow c \rightarrow s$

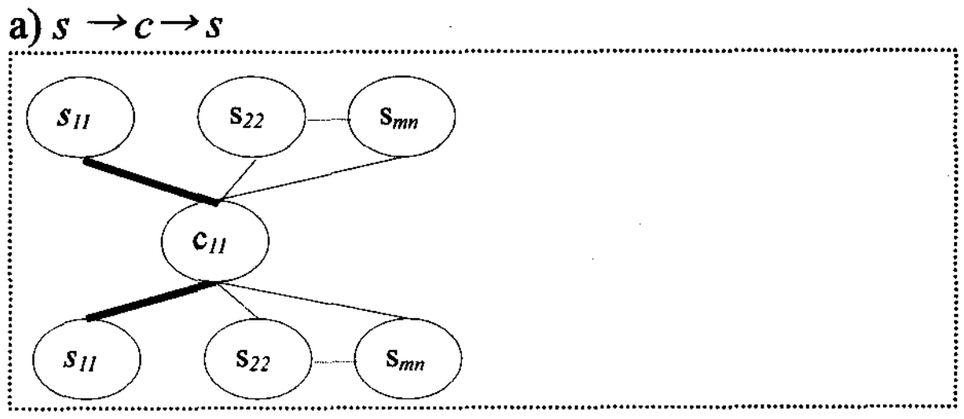


Fig. 6 Dependence tree with data association (*s,c,s*)

b) $s \rightarrow c \rightarrow C$

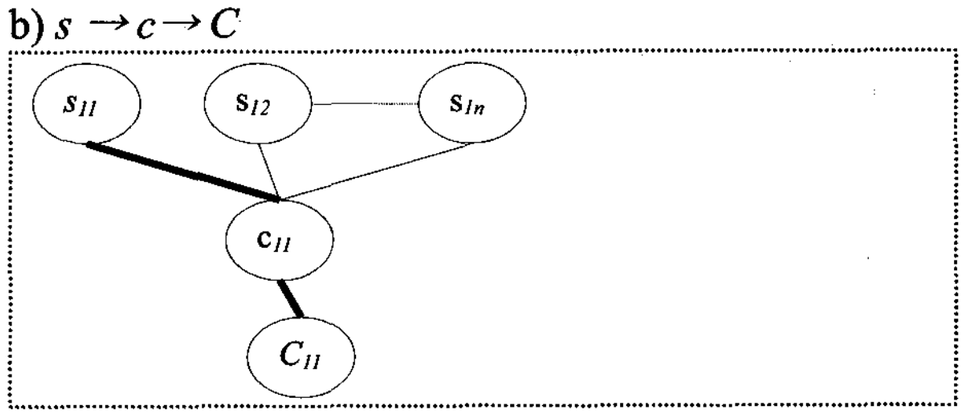


Fig. 7 Dependence tree with data association (*s,c,C*)

c) $C \rightarrow s \rightarrow c$

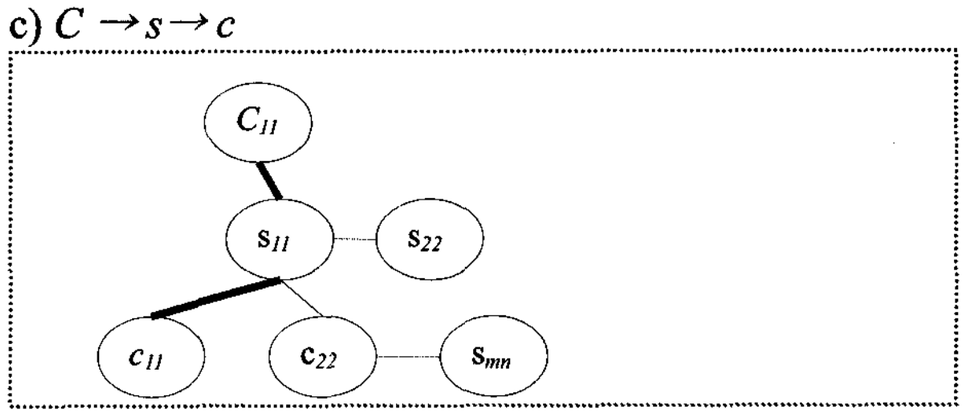


Fig. 8 Dependence tree with data association (*C,s,c*)

We can detect specific sequences of events from this dependence tree. The sequence of event Seq^{+} (*s*) indicates that an event s_{11} is followed by an event s_{12}, which is followed by an event s_{13}, and so on.

$Seq^+(s) = (s_{11}, s_{12}, \ldots, s_{1n})$
$Seq^+(c) = (c_{11}, c_{12}, \ldots, c_{1n})$
$Seq^+(C) = (C_{11}, C_{12}, \ldots, C_{1n})$

This basic sequence of events can extend to take composite events as shown above the dependence trees with data association which can be accomplished by running below Algorithm. Below algorithm illustrates the process of building the dependence tree.

Algorithm BuildDependenceTree

Input: Single events *s*, Composite events *c* and *C*
Output: Dependence tree
Method:

```
1: begin
2:    for each event s(i) do
3:       if event c == event i+1 and
4:          s(Pid) == c (Pid) then
5:          link s(i) and c(i+1)
6:          if event s or C == event i+2 and
7:          s(Pid) == C (Pid) then
8:          link c(i+1) and (s(i+2) or C(i+2));
9:       endif
10:   endfor
11:   for each event C(i) do
12:      if event s == event i+1 and
13:         C(Pid) == s (Pid) then
14:         link C(i) and S(i+1)
15:         if event c == event i+2 and
16:         s(Pid) == c(Pid) then
17:         link s(i+1) and c(i+2);
18:      endif
19:   endfor
20:   return dependence tree
21: end
```

IV. EXPERIMENTAL EVALUATION

We experimentally evaluate the dependence tree with data association technique based on state charts of temporal events. We use the experimental parameters as follows: historical queries for event tracking, number of events, range of linked events, and page size. We randomly place uniformly number of linked events between 1 and 10. For the experiments we assumed that we have a page size of 4KB and generated 100 random tracking queries. All experiments were conducted on an Intel Pentium 4 3.0 GHz and Windows XP.

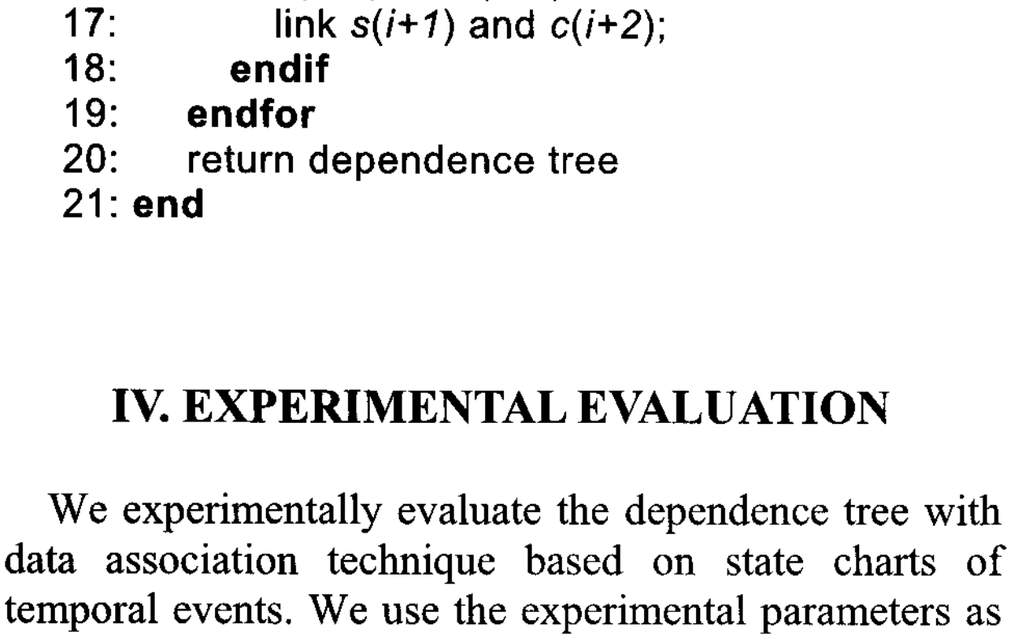


Fig. 9 Linked events

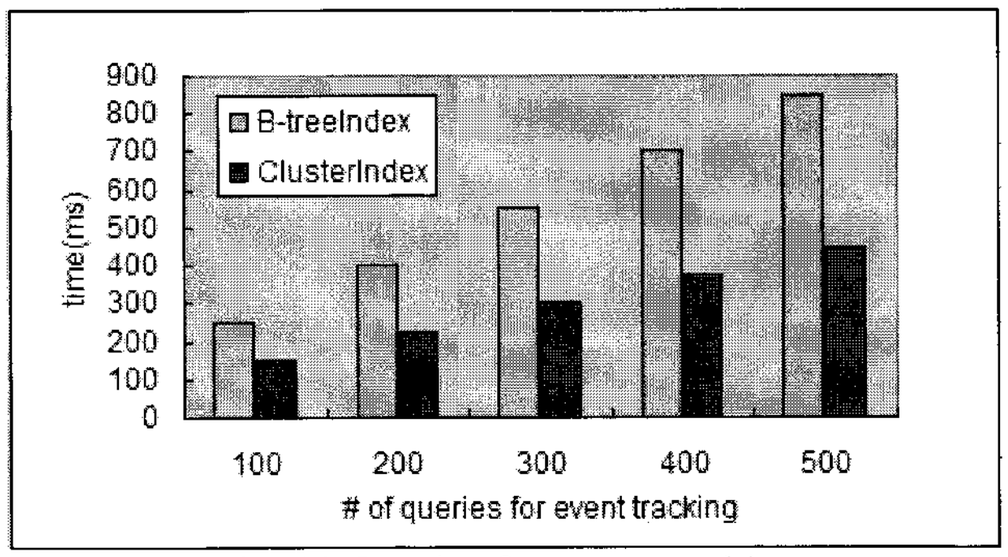


Fig. 10 Queries for event tracking

Fig. 9 shows the average response time compared with the B-tree index and the cluster index based on dependence tree. In this case we vary the number of linked events from 1 to 10. The cluster index based on dependence tree has a fast response time around much more 2 times the B-tree index when number of linked events is 10 as shown in Fig9. The difference between the B-tree and the cluster is almost seek time for event tracking, for less seek time the advantage of using the dependence tree with data association.

Fig. 10 also shows the effect of dependence tree on query processing. The cluster index base on the dependence tree is significantly faster than the B-tree index. This is expected as the B-tree uses more blocks for queries processing than the cluster index with linked events. The cluster index based on dependence tree with data association benefits from using very short seeking.

V. CONCLUSIONS

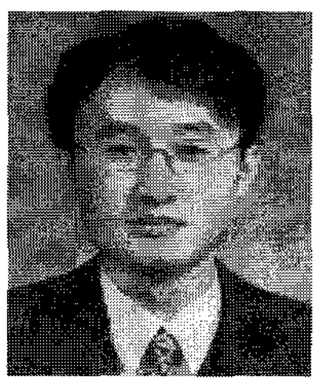
One of the biggest issues in RFID is to deal with a huge amount of data. The volume and velocity of RFID data will exceed the capacity of existing technology infrastructure. It could incur slower queries and updates. We have proposed a clustering technique for fast query processing. Our study considers two types of events that are single events and composite events with temporal pattern. We classified such events and then present the state charts of temporal event flow. Also we propose the dependence trees with data association and use them to cluster the linked events in which to execute rules.

According to our experimental evaluation, the cluster index based on dependence tree has a fast response time on varying number of linked events. In case of varying number of queries event tracking, also the cluster index outperforms the B-tree index. We show the power of our clustering technique based on proposing the dependence tree in efficient response of event tracking queries.

REFERENCES

[1] B. Glover and H. Bhatt, *RFID Essentials*, O'Reilly Media, Firtst Edition, 2006, pp. 2–3.
[2] A. Asif and M. Mandviwalla, "Integrating the supply chain with RFID: A technical and business analysis," *Communications of the Association for Information*

Systems, vol. 15, 2005, pp.393-427.
[3] Siemans to Pilot RFID Bracelets for health Care. Available: http://www.infoworld.com/article/04/07/23/HNrfidimplants_1.html, July 2004.
[4] H. Gonzalez, J. Han, X. Li, and D. Klabjan, "Warehousing and Analyzing Massive RFID Data Sets," 22nd IEEE ICDE Conference, 2006, p.1.
[5] Y. Bai, F. Wang and P. Liu, "Efficiently Filtering RFID Data Streams," CleanDB, 2006, pp. 50–57.
[6] K. Finkenzeller, *RFID Handbook: Fundamental and Applications in Contactless Smart Cards and Identification*, John Wiley & Sons, 2003.
[7] C. Heinrich, *RFID and Beyond. Growing Your Business Through Real World Awareness*. Wiley Publishing, Indiana, USA, 2005.
[8] R. Derakhshan, M. Orlowska, and X. Li, "RFID Data Management: Challenges and Opportunities," IEEE International Conference on RFID 2007, Texas, USA, 2007, p.176.
[9] EPC tag data standard version 1.1., EPC Inc, June, 2005.
[10] S. Jeffery, M. Garofalakis, M. Franklin, "Adaptive Cleaning for RFID Large Data Bases," 32nd International Conference on VLDB, 2006, pp.163-174.
[11] M. Oleksandr. Available: http://epic.hpi.uni-potsdam.de/pub/Home/Publications/RFID-Paper_SS2007_Oleksandr_Mylyy.pdf
[12] F. Wang and P. Liu, "Temporal Management of RFID data," Proceeding of the VLDB05, 2005, pp.1128-1139.
[13] M. Palmer, *Principles of Effective RFID Data Management*. Enterprise Systems, 2004.
[14] S. Laurie, *RFID Implementation Challenges Persist, All This Time Later*. Information Week, Oct. 2005.
[15] L. Shaorong, W. Fusheng and L. Peiya, "A Temporal RFID Data Model for Querying Physical Objects," A Time Center TR-88, Feb. 2007, pp. 6-8.



Hong-won Yun
He received his B.S. and the Ph.D. degrees at the Department of Computer Science from Pusan National University, Korea, in 1986 and 1998, respectively. He is a professor at the Department of Information Technology, Silla University in Korea. His research interests include temporal database and data stream management.