Adaptive Partitioning for Efficient Query Support

Hong-won Yun, *Member, KIMICS*

Abstract—RFID systems large volume of data, it can lead to slower queries. To achieve better query performance, we can partition into active and some non-active data. In this paper, we propose two approaches of partitioning for efficient query support. The one is average period plus delta partition and the other is adaptive average period partition. We also present the system architecture to manage active data and non-active data and logical database schema. The data manager check the active partition and move all objects from the active store to an archive store associated with an average period plus data and an adaptive average period. Our experiments show the performance of our partitioning methods.

Index Terms—Partitioning methods, Data archiving, RFID data management, Temporal data modeling.

I. INTRODUCTION

RFID (Radio Frequency Identification) technology can be used to improve the efficiency of business processes by providing the capability of automatic identification and data collection. It can achieve greater visibility and product velocity across supply chains, easier product tracking and monitoring and much more labor cost. Despite the diversity of RFID applications, RFID data have to be considered in RFID data management systems [1-4]. In current RFID applications while the accuracy of current RFID sensors is improving, there are erroneous readings. Such duplicate readings and missing readings data are filtered semantically [5,6,12].

RFID data are generated quickly and accumulated for querying. Very large data volume need to store RFID data and requires a scalable storage scheme to assure efficient queries and updates. Handling historical data we had initially raised about data volume generated by an RFID system leads to a second and equally important issue [7-9]. If just one of your customers relies on RFID and asks you about a specific item by unique ID then the system has to be able to get the information by that identification. Equally important are cases where you have product recall or returns management [4].

Manuscript received November 23, 2007.

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)

RFID systems generate large volume of data with fast reading speed. Very large volume of data can cause of slower queries and updates. Generally RFID data have limited active life span during which the data are updated and queries such as tracking and monitoring [1,12-14]. An object starts from the time when it is first tagged, moved, read and ends when the object is sold to customers. RFID data can be partitioned into active data and non-active data. This partitioning can perform most queries and do updates of data efficiently.

In this paper, we propose two approaches of partitioning for fast querying speed with RFID data streams. The rest of the paper is organized as follows. Section II presents the RFID system architecture to treat large volume of data and logical database schema. In Section III we propose two partitioning methods that are the average period plus delta partition and the adaptive average period partition. Section IV shows the experimental results. Finally we conclude our study in Section V.

II. SYSTEM ARCHITECTURE

A. RFID System

The RFID system in this paper consists of the following components: RFID tags, Readers, Stream Manager, and Data Server. The Data Server includes Data Manager, Data Store, Data Archive, and Product Data Store. Each component's function is as followings.

RFID tags. RFID tags can be attached to or embedded in a physical object to be identified. They store product-item information such as manufacturer, product lot, size, production date, expiration date, etc. RFID tags come in a large variety of designs and have many different functional characteristics. EPCglobal classifies RFID tags into 5 classes. We use Class 0, 1, and 2 tags in this paper.

RFID readers. RFID readers are electronic devices that emit and receive radio signals through the antennas coupled to them. Readers are responsible for the information flow between the tags and the Data Server via the Stream Manager.

Stream Manager. Stream Manager receives data from readers, filter the data, and send the data to the Data Server. Each Stream Manager can connect to multiple readers and process the data generated from the readers. The data filtering is essential function of the Stream Manager. The filtering functions are to preliminarily screen raw reading data and can be

removing duplicate data, error detection, and so on.

Data Manager. Data Manager is at the core of an RFID system. It is responsible for expressive data modeling, querying including object tracking and monitoring, and data migration. Based on the temporal data model, common RFID queries can be effectively supported. The data migration is its key function to move RFID data from Data Store to Data Archive in this paper.

Data Store. Data Store stores active RFID data for querying including object tracking and monitoring and provides logical database schema.

Data Archive. Data Archive is non-active data storage. Non-active data are moved and archived into historical partitions.

In this paper, this RFID system architecture is designed essentially for active and non-active data management shown in Fig.1.

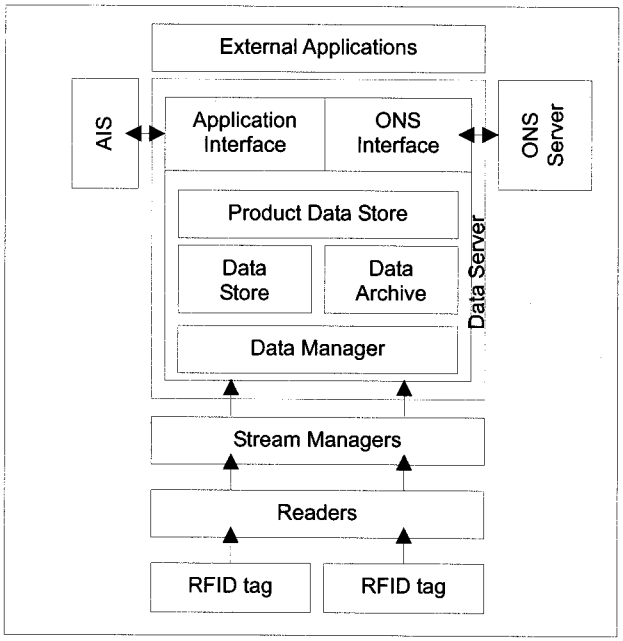


Fig. 1 RFID system architecture

B. Logical Database Scheme

This paper focuses on a single loop supply chain in the retail industry to explore issues related to the integration of RFID system and the EPC Network between different partners. The EPC code offers the means for unique identification of any object throughout a supply chain. Once an EPC code is incorporated into an RFID tag and attached to a physical object, the object become unique in the world. Assume that we record each item movement with a tuple of the form, (EPC, name, time) where EPC uniquely identifies each item. Among all the data, they dynamically interact with each other and generate business logic. These interactions generate events and states changes. Such changes need to develop new data model that may provide fast response to such queries.

There can be many entities in RFID applications. We consider as fundamental entities in RFID applications. These involve objects, readers, loca-tions, and transactions. Relationships are generated when entities interact with each other. Generally entities are related to static data such as object name, location information. Relationships are related to dynamic data such as location and changes of objects. The temporal data are directly related to these RFID applications. There are two types of temporal elements that are event times and state times. We use two attributes event start time and event end time are associated with a static data; and state start time and state end time are associated with a dynamic data. When an reader start up at the specific location, the reader out of order after a period of time, then a single tuple of the form (reader_epc, reader_name, event_start_time, event_end_time) will be generated as shown in Fig.2.

reader_epc	reader_name	event_start_time	event_end_time
1.1.255.1	Reader1	2005-10-10 10:00:00.000	2007-10-11 10:00:00.000
1.1.255.2	Reader2	2005-10-12 10:00:00.000	Now
1.1.255.3	Reader3	2005-10-15 11:00:00.000	2007-03-20 10:00:00.000
1.1.255.4	Reader4	2007-10-25 11:00:00.000	Now

Fig. 2 Sample event time

When an object stays at the same location, the object will be leaved the location after a period of time, and then a single one of the form (epc, location_id, state_start_time, event_end_time) will be registered as shown in Fig.3.

epc	location_id	state_start_time	state_end_time
1.1.1.1	Loc001	2007-10-10 10:00:00.000	Now
1.1.1.2	Loc002	2007-10-12 10:00:00.000	2007-10-12 15:00:00.000
1.1.1.3	Loc003	2007-10-15 11:00:00.000	Now
1.1.1.4	Loc004	2007-10-25 11:00:00.000	2007-10-28 12:00:00.000

Fig. 3 Sample state time

III. PARTITION METHODS

A. Average Period plus Delta Partition

Stream Manager with filtering function generates large volume of data and Data Manager stores all data in Data Store at first. We assume all active objects are stored in Data Store which has an active partition P_k. Data Manager check the active partition for every fixed period and move all objects from the active partition to an archive partition. Moving objects have their end time less than the end time of fixed period. Non-active objects will be moved into Data Archive (P_1, P_2, ...) for every fixed period as

shown in Fig. 4 (dotted lines represent the life spans of non-active objects).

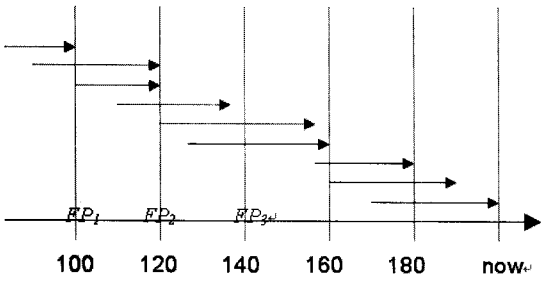


Fig. 4 Fixed period partition

The active object set is defined as:

$$AOS_i = \{E_{ij} \mid \text{now-}\ FP_i \leq E_{ij}.ST \quad \}$$

where E_{ij} is the object id, $E_{ij}.ST$ is the event start time or the state start time of each object, and FP_i is the fixed period for an active partition. The non-active object set is defined as:

$$NAOS_i = \{E_{ij} \mid E_{ij}.ET < \text{now-}\ FP_i \}$$

where $E_{ij}.ET$ is the event end time or the state end time of each object. We suppose two temporal attributes, $E_{ij}.ST$ is the event start time or the state start time of each object, $E_{ij}.ET$ is the event end time or the state end time of each object, and then a period of an object is $l_{11} = E_{11}.ET - E_{11}.ST$. A set of period for all objects L is as following: $L = \{l_{11}, l_{12}, \ldots, l_{mn}\}$. The number of periods for L is $n = |L|$. The average period of all objects AP is defined as:

$$AP = \frac{\sum_{l_{ij} \in L} l_{ij}}{n}$$

A set of P_1 is an object set including periods of objects, which are included the period $(AP_1 + \Delta)$. The Δ is a time granularity more than second. The last partition P_k has active objects that are defined as follows:

$$P_k = \{E_{ij} \mid \exists E_{ij} \in AOS_i \text{ such that } l_{ij} < AP \}$$

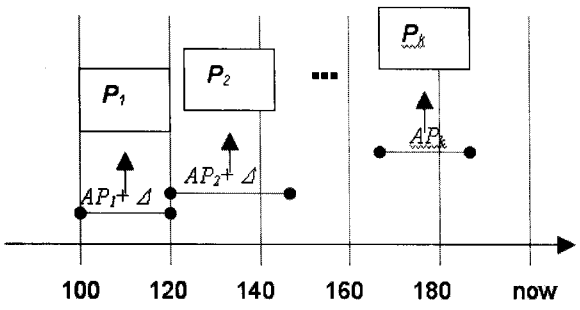


Fig. 5 Average period plus delta partition

B. Adaptive Average Period Partition

When the object life spans are regular and not with long lived objects, the above fixed period partition and Average Period plus Delta partition works well. Also, when almost of queries are performed on active partition, these both partition methods are efficient. There are various RFID applications and the object life spans are not uniform. We have common RFID queries such as object tracking and monitoring, and other complex queries e.g., time slicing and snapshot queries, these queries should be effectively supported on the proper search spaces.

On the other hand, lifespan of event-based objects are short, state based objects generate dynamic state histories these lifespan are long. Although a state-based object becomes non-active data having end time value, if this object still stays on other tables, for example, not sold out to customer, the object is not moved to Data Archive and be kept in Data Store P_k as following form:

$$P_k = \{E_{ij} \mid \exists E_{ij} \in AOS_i \text{ such that } (l_{ij} < AP \wedge NAOS_{i-1})\}$$

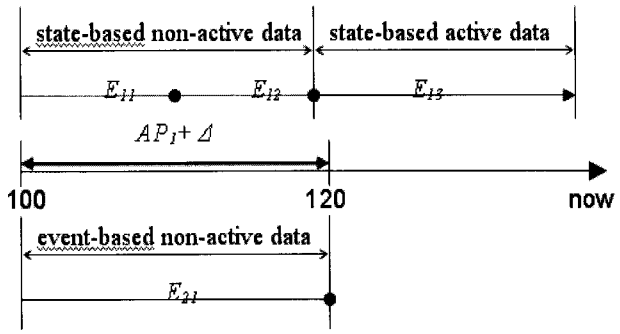


Fig. 6 Adaptive average period partition

Here we use dash-dot lines to represent non-active data, and dash lines to represent active data in Fig. 6. It may be possible to move non-active data using average period plus delta that presented in Algorithm 1. Delayed data moving by some possibility can be accomplished by running Algorithm 2.

Algorithm 1 AveragePeriodDeltaPartition

Input: active object E, i^{th} average period AP_i
Output: non active partition P_i
Method:
1: **begin**
2: **for** each active object E in data_store
3: **if** $E_{ij}.ET <$ now- $(AP_i + \Delta)$ **then**
4: move E_{ij} in P_i in
5: **else** stays E_{ij} in data_store
6: **end for**
7: move P_i into archive_store
8: **end**

Algorithm 2 AdaptiveAveragePeriodPartition

Input: active object E, i^{th} average period AP_i
Output: non active partition P_i
Method:
1: **begin**
2: **for** each active object E in data_store
3: **if** $E_{ij}.ET <$ now- $(AP_i + \Delta)$ **then**
4: **if** $E_{ij+1} ==$ null **then**
5: move E_{ij} in P_i
6: **else** stays E_{ij} in data_store
7: **else** stays E_{ij} in data_store
8: **end for**
9: move P_i into archive_store
10: **end**

IV. PERFORMANCE STUDY

In this section, we perform a thorough evaluation of our partitioning methods. The experiments were implemented using C++ and were conducted on an Intel pentium4 3.0GHz system with 2GB of RAM. We use randomly event-base queries and state-based queries based on historical and current data to evaluate performances. We use the following parameters and values.

Parameter	Value
Number of data	100,000
Data size	256 byte
Number of partitions	30
Time period	100, 110, ..., 200
Rate of non-active data	10,20, ..., 50%

Fig. 7 shows the average response time of the adaptive average period partition and the average period plus delta partition compared with the fixed period partition. The adaptive average period partition has a fast response time around 1.5 times the fixed period partition when the average interarrival time is 100 milliseconds as shown in the Fig 7. As expected the size of the adaptive average period partition at state-based queries increases adaptively.

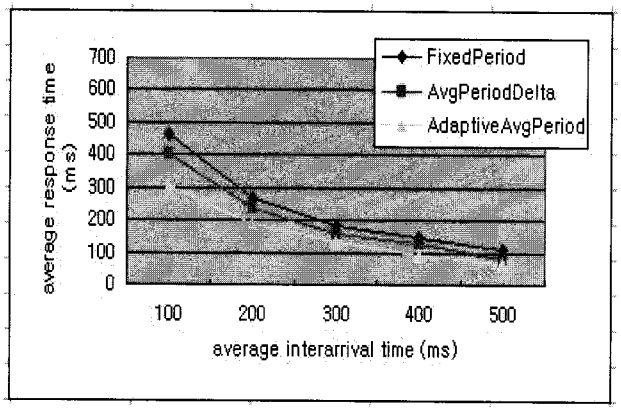


Fig. 7 Average response time vs. Average interarrival time

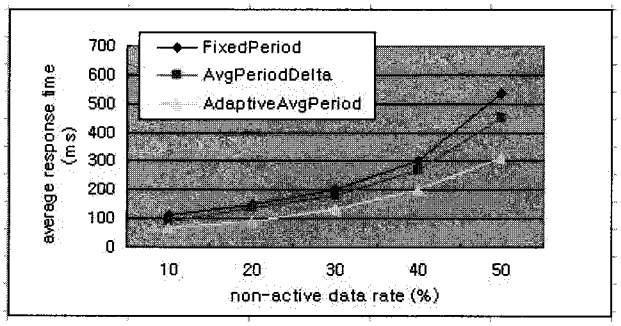


Fig. 8 Average response time vs. Non-active data rate

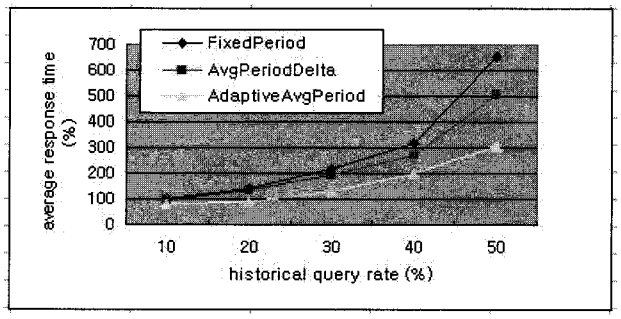


Fig. 9 Average response time vs. Historical query rate

Fig. 8 also shows the average response time of the adaptive average period partition and the average period plus delta partition compared with the fixed period partition. In this case we vary the rate of non-active data from 10% to 50 %. As it can be seen in the figure, the adaptive average period partition clearly outperforms the fixed period partition. This is expected as the fixed period partition uses more data blocks for queries than the average period plus partition and the adaptive average period partition.

Fig. 9 shows the response time on varying of historical query rate. The query executing on the adaptive average period partition is significantly faster than the fixed period partition. We can see that the rate of temporal range queries come into effect the difference between the fixed period partition and the average period partition series. The range queries on the fixed period partition need more data blocks to execute than the adaptive average period partition.

A major contribution of the proposed partition methods is the ability to efficiently response time at various experiments.

V. CONCLUSIONS

RFID data are generated quickly, accumulated for querying and become large volume of data. It can lead to slower queries and updates. We can partition into active data and non-active data to achieve better query performance and efficient updates. The fixed period partition is well known to us. In this paper, we propose two methods of partitioning for efficient query support: average period plus delta partition and adaptive average period partition. The data manager in the RFID system check the data store and move all objects from the active

store to an archive store associated with an average period plus data and an adaptive average period. We present the system architecture to manage active data and non-active data and logical database schema to implement data stream database. We experiment to show the performance of two our partitioning methods and the adaptive average period algorithm clearly outperforms the other two methods. We believe that further study is needed to implement the data manager for these partitioning methods.

REFERENCES

[1] S. S. Chawathe, V. Krishnamurthy, S. Ramachandrany, and S. Sarma. "Managing RFID Data," VLDB, pp.1189-1195, 2004.

[2] M. Palmer. "Seven Principles of Effective RFID Data Management," www.objectstore.com/docs/ articles/7principles_rfid_mgmnt.pdf, Aug. 2004.

[3] S. Liu, F. Wang and P. Liu, "Integrated RFID Data Modeling: An Approach for Querying Physical Objects in Pervasive Computing," CIKM'06, Nov. 2006.

[4] EPCglobal. The EPCglobal Network, 2004. Available: http://www.epcglobalinc.org

[5] Y. Bai, F. Wang and P. Liu, "Efficiently Filtering RFID Data Streams," CleanDB, Sep. 2006.

[6] S. Sarma, "Integrating RFID," *ACM Queue*, 2(7), pp.50-57, October 2004.

[7] A. Asif and M. Mandviwalla, "Integrating the supply chain with RFID: A technical and business analysis," *Communications of the Association for Information Systems*, 15, pp.393-427, 2005.

[8] Thomas Diekmann, Adam Melski, and Matthias Schumann, "Data-on-Network vs. Data-on-Tag: Managing Data in Complex RFID Environments," 40th HICSS'07, pp.224-234, 2007.

[9] Fosso Wamba et al., "Enabling Intelligent B-to-B eCommerce Supply Chain Management using RFID and the EPC Network: a Case Study in the Retail Industry," *International Journal of Networking and Virtural Organizations*, 3(4), pp. 450-462, 2006.

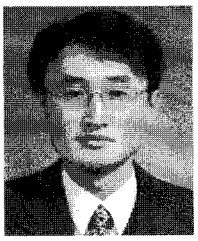
[10] Boris Bonfils and Philippe Bonnet, "Adaptive and decentralized operator placement for in-network query processing," IPSN2003, pp.1361-1364, April 2003.

[11] V. D. Berg, J. P. and W. H. M. Zijm, "Models for Warehouse Management: Classification and Examples," *International Journal of Production Economics*, 59, pp. 519-528, 1999.

[12] Bai, Y., Wang, F., Liu, P., "Efficiently Filtering RFID Data Streams," In CleanDB Workshop, pp. 50-57, 2006.

[13] Gonzalez, H., Han, J., Li, X., Klabjan, D., "Warehousing and Analyzing Massive RFID Data Sets," 22nd IEEE ICDE Conference, 2006.

[14] Jeffery, S., Garofalakis, M.,Franklin, M.: Adaptive Cleaning for RFID Large Data Bases, 32nd international conference on VLDB, pp.163-174, 2006.



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.