ISSN:0975-9646

An Adaptive Representation of RFID Data Sets Based on Movement Graph Model

M. P. Ravikanth, A. K. Rout
CSE Department, GMR Institute of Technology, JNTU Kakinada, Rajam

***Abstract*— Radio Frequency Identification (RFID) data sets are expected to become commonplace in supply chain management systems. This model tells about movement graph model as a compact representation of RFID data sets. Since the information of item is associated with the objects as well as spatiotemporal. The movement graph can be huge, complex, and multidimensional in nature. RFID used for identifying, locating, tracking and monitoring physical objects. To achieve these goals RFID data have to be collected, filtered and transformed into semantic application data. Such data cannot be used directly by applications unless they are filtered and cleaned.**

The Movement graph can be effectively organized by the gateway nodes, which serve as bridges for connecting different regions of the graph. We propose an efficient cubing algorithm that performs simultaneous aggregation of both spatiotemporal and item dimension on a partitioned movement graph. Main goal of this research project is to design and develop an efficient, RFID processing system that enables real-time tracking monitoring.

Keywords— **RFID, data warehousing, data models**

I. INTRODUCTION

RFID (Radio Frequency Identification) is a non-contact automatic identification technology, which aims at identifying and tracking items by using radio frequency electromagnetic wave to let readers capture the data on RF tags attached to them.

However, unreliable data (original data) captured by readers is a major factor hindering the development of RFID technology. Under normal circumstances, it is quite often that the loss and error rate is between 30-40 %.

For effectively and efficiently supporting high-level RFID business logic processing, it is necessary to provide high-quality RFID data. For that case, it is critical to clean the original data.

One major problem to be solved in pervasive computing is to identify and track physical objects, and RFID technology is a perfect fit to solve this. By tagging objects with EPC 1 tags that virtually represent these objects, the identifications and behaviours of objects can be precisely observed and tracked.RFID readers can be deployed at different locations and networked together, which provides an RFID-based pervasive computing environment.

The filtered RFID data often need to preserve the original order, i.e., the first observed tagged object will be output first after filtering. Such order can be critical for many RFID applications.

II. RADIO FREQUENCY IDENTIFICATION (RFID)

Technology that allows a sensor (reader) to read, from a distance, and without line of sight, a unique electronic product code (EPC) associated with a tag.

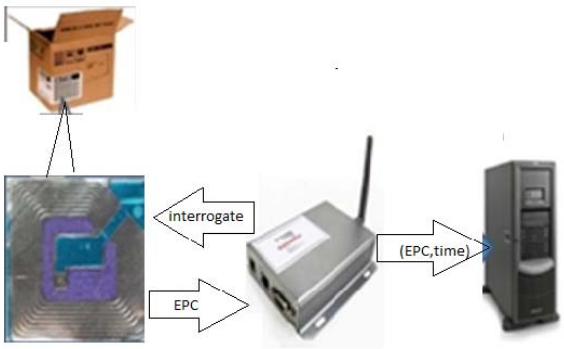

Fig. 1 Radio Frequency Identification

III. RFID FRAMEWORK ARCHITECTURE

It is one of the first RFID integration platforms focusing on large-scale deployments. Its service-oriented architecture provides network services to applications through several standard protocols and interfaces. Java System RFID Software consists of two major components: The event manager processes (filters and aggregates) RFID data, while the information server provides access to the business events generated by the event manager and serves as an integration layer that offers options for integrating with enterprise applications.

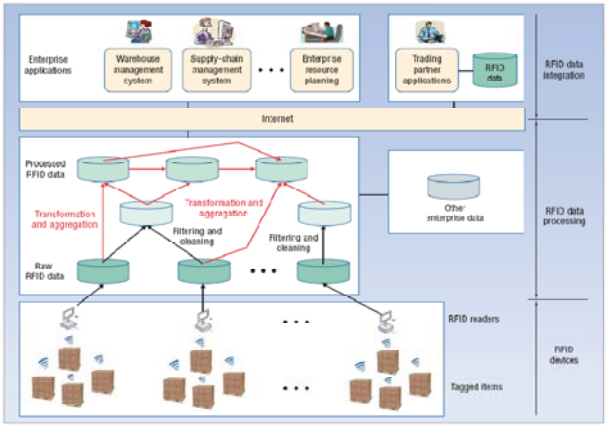

Fig. 2 RFID Frame Work Architecture

It is one of the first RFID integration platforms focusing on large-scale deployments. Its service-oriented architecture provides network services to applications through several standard protocols and interfaces. Java System RFID Software consists of two major components: The event manager processes (filters and aggregates) RFID data, while the information server provides access to the business events generated by the event manager and serves as integration layer that offers options for integrating with enterprise applications.

IV. PROBLEM DEFINITION

In order to realize the full benefits of detailed object tracking information, we need to develop a compact and efficient RFID cube model that provides OLAP-style operators useful to navigate through the movement data at different levels of abstraction of both spatiotemporal and item information dimensions.

This is a challenging problem that cannot be efficiently solved by traditional data cube operators, as RFID data sets require the aggregation of high-dimensional graphs representing object movements, not just that of entries in a flat fact table.

V. PROPOSAL STATEMENT

In paper will be developed on effective and efficient RFID data filtering techniques to generate clean RFID data, which can be further interpreted and integrated into RFID-based applications.

In this paper, two types of filtering is proposed: noise is removed from RFID data (de-noising or smoothing), and duplicates are merged into one distinct reading (duplicate elimination, or merging).

VI. APPROACH/METHODS

The work proposed here to model the RFID data warehouse using a movement graph-centric view, which makes the warehouse conceptually clear, better organized, and obtaining significantly deeper compression and performance gain over competing model in the processing of path queries.

The path databases used for performance evaluation were generated using a synthetic path generator. Locations inside a partition are arranged according to a producer configuration, where we simulate factories connecting to intermediate locations that aggregate traffic, which in turn connect to Out-gateways; or a consumer configuration, where we simulate products moving from In-gateways, to intermediate locations such as distribution centers, and finally to stores.

Generate paths by simulating groups of items moving inside a partition, or between partitions, and going usually through gateways, but sometimes, also “jumping” directly between non-gateway nodes.

Construction of a RFID cube by Applying Following Techniques.

1. Movement Graph Aggregation.
2. Generate RFID Cuboids.
3. Cube Computation.

VII. MOVEMENT GRAPH PARTITIONING

Gateway Identification:

a. First Procedure

Based on the paths, we construct a movement graph, which are partitioned along gateway nodes Generally ,in supply chain management data analysts provides the complete list of gateways.

b. *Second Procedure*

Find the edge, if that edge is removed from the graph then the graph splits into two disconnected components.

c. Third Procedure

Based on the large traffic flow through nodes and they can also be identified by using a concept of betweens and centrality in social network analysis.

Algorithm been used to find the Gateway node is Depth First Search (Backtracking is possible).

ALGORTHIM TO CREATE A GRAPH

Step: 1 begin
Step :2 Read no of locations
Step :3 Read the adjacency among locations (0 or 1) in an array x[i][j] .
Step :4 Initially all locations are un-visited i.e visited[I]<-
False.
Step:5 print x[i][j].

ALGORITHM TO CREATE DFS EXCLUDING MISSING NODE

Step:1 Begin.
Step:2 call function dfs(curv,missv) where initially curv<-i ,missv<-i.
Step:3 find whether the node is visited or not , if the node is not visited then perform visited[curv]=true.

Step:4 Intialize k<-0 increment k by 1 until k <n(no of locations).
Step:5 Verify the condition such that if(k!<-missv && k!<-curv && x[curv][k]<-1 && !visited[k]) visited[k]<-true.Now invoke recursive function dfs(k,missv) if the condition fails goto step3.
Step:6 Now goto step2 and continue the process.
Step:7 Stop

ALGORITHM TO FIND GATEWAYS

Step: 1 Begin.
Step: 2 Initialize i<-0,j<-0 where visited[j]=false.
Step: 3 Initialize curv<-I,missv<-I
Step: 4 Generate Randomly a no to visit the intial location
Now ,find if(missv!=curv)
break;//comes out of loop.
Step: 4.1 Call fun dfs(curv,missv)
Initialize j<-0 ,j<-j+1,until j<n;
check whether if(j!=missv && !visited[j])
//if condition is true break.
Step: 5 Check if(j!=n)//if condition is true add gateway to the vector.

VIII. RFID DATA GENERATION

Raw RFID data consists of a set of triples (TagID, Loc, Time), where TagID is the Electronic Product Code (EPC) of the tag and is used for identifying the tag uniquely.Loc is the location of the RFID reader which detects the tag.Time is the time of detecting the tag.

We translate raw RFID data generated in supply chain management into a set of stay records that do not have duplicates. A stay record has the form (TagID, Loc, Start Time, End Time),where TagID and Loc are the same as above.

Start Time is the time when the tag enters the location.
End Time is the time when the tag leaves the location.
From the stay records of a tag, we can construct the trace record of the tag in the form of

$$TagID: L1[S1, E1] \rightarrow \cdots \rightarrow Ln[Sn, En]$$

where L1. . .Ln are the locations where the tag is detected, Si is Start Time at the location Li, Ei is End Time at the location Li, and Li[Si;Ei] is ordered by Si. We use a set of trace records instead of raw RFID data in our systems.

(tag1,A,2),(tag4,A,2),(tag2,A,2),(tag3,A,2),(tag1,A,3),
(tag2,A,3),(tag4,A,3),(tag3,A,3),(tag3,B,5),(tag1,B,5),
(tag2,B,5),(tag4,B,5),(tag1,B,6),(tag4,B,6),(tag3,B,7),
(tag1,B,7),(tag2,B,7),(tag4,B,7),(tag2,C,8),(tag1,C,8),
(tag3,C,8),(tag3,C,9),(tag1,C,9),(tag2,C,9),(tag4,C,13),
(tag4,C,14),(tag4,C,16)

Fig. 3 Raw Data

Tag1:A[2,3]->B[5,7]->C[8.9]
Tag2:A[2,3]->B[5,7]->C[8,9]
Tag3:A[2,3]->B[5,7]->C[8,9]
Tag4:A[2,3]->B[5,7]->C[13,16]

Fig. 4 Trace Records

IX. QUERIES TEMPLATES FOR OBJECT TRANSITION.

The tracking query finds the movement history for the given tag. The path oriented query is classified into the path oriented retrieval query and the path oriented aggregate query. The path oriented retrieval query finds tags that satisfy given conditions (including a path condition) and the path oriented aggregate query computes the aggregate value for tags that satisfy given conditions (including a path condition).

Following Figure shows the formal definition for query templates in supply chain management.

[1] Tracking Query=<TagID=ID>
[2]Path Oriented Retrieval Query =<Path Condition , Info Condition>
[3]Path Oriented Aggregate Query = <Aggregate Function, Path Condition, Info Condition>
Path Condition → (Step)*
Step → Loc[Time Condition] | //Loc[Time Condition]
Aggregate Function →count( )| sum(Time Selection) | avg (Time Selection) | max(Time Selection)
| min (Time Selection)
Time Selection →Loc Start Time -Loc End Time |
Loc End Time-Loc Start Time
---
*Info Table has the information for the tags such as product name, manufacturer, and price.
** Time Condition is the predicate for start and end time.
***Loc is the location name of a detection region.

Fig. 5 Query Templates for Tracking Queries and Path Oriented Queries

Semantics	Query
Find the movement history for the tag whose identifier is XYZ(Tracking Query)	TagID=XYZ
Find the tags that go through locationL_1—L_2 (Path Oriented Query)	<//L_1//..L_2>
Find the tags that go through locations L_1—L_2 where the duration at L_2 is <T(Path Oriented Retrieval Query)	<//L_1[(End Time-Start Time) <T]//...L_2>
Find the average duration time at L_2 for tags that go from L_2 directly to L_2(Path Oriented Aggregation Query)	<avg(L_2.EndTime-L_2.StartTime), //L_1//L_2>
Find the minimum start time at L_2 for laptops that go from L_1 to L_2 (Path Oriented Aggregation Query)	<min(L_2.Start Time),//L_1/L_2, Product name =”laptop”.

Fig. 6 Examples for Tracking Queries and Path Oriented Queries

X. ARCHITECTURE

Following figure shows the architecture to store RFID data, and process tracking queries and path oriented queries in supply chain management. The central server receives raw RFID data from various regions whose format is (*TagID; Loc; Time*). The raw RFID data is transformed into trace records after sorting the RFID data by the tag identifier and the time (i.e., *TagID* : $Loc_1[S_1;E_1]$- - > Loc_n $[S_n;E_n]$). The path information in the trace records is stored by using Element List Encoding Number and Order Encoding Number and the time information in the trace records is stored by using Region Number. Since we use prime numbers instead of location names, the (Location, Prime Number) list is kept in memory as a hash structure. Based on the above encoding schemes, we store trace records by using the relational schema (PATH TABLE, TAG TABLE, and TIME TABLE). If a user requests a tracking query, a path oriented retrieval query, or a path oriented aggregate query, Query Translator translates it into an SQL query. Then, the SQL query is processed by an RDBMS and the result is sent to the user.

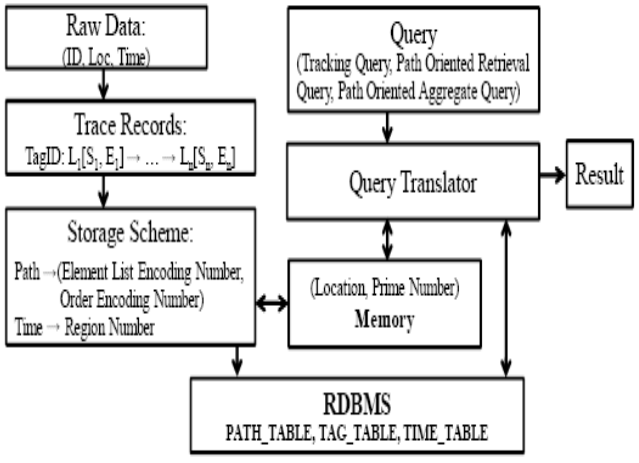

Fig. 7 Architecture to store RFID data, and process tracking queries and path oriented queries in supply chain management.

XI. RESULTS

In this study, we develop a movement graph model as a compact representation of RFID data sets. Since spatiotemporal as well as item information can be associated with the objects in such a model, the movement graph can be huge, complex, and multidimensional in nature. We show that such a graph can be better organized around gateway nodes, which serve as bridges connecting different regions of the movement graph.

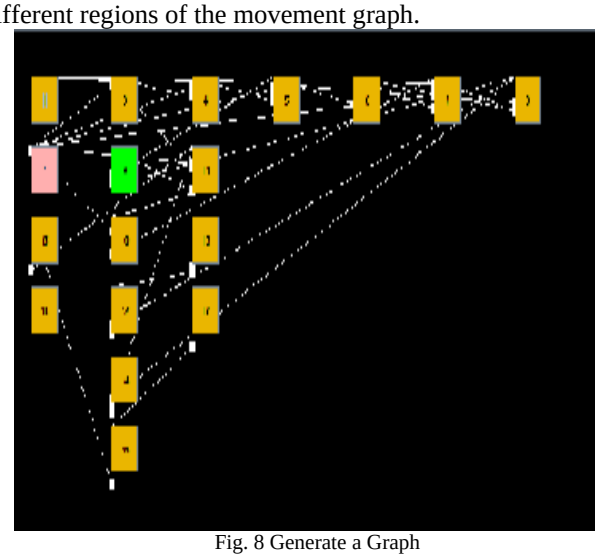

Fig. 8 Generate a Graph

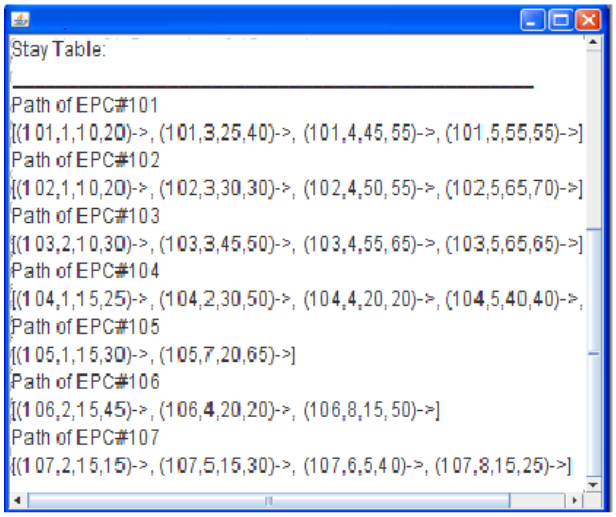

Fig. 9 Stay Table.

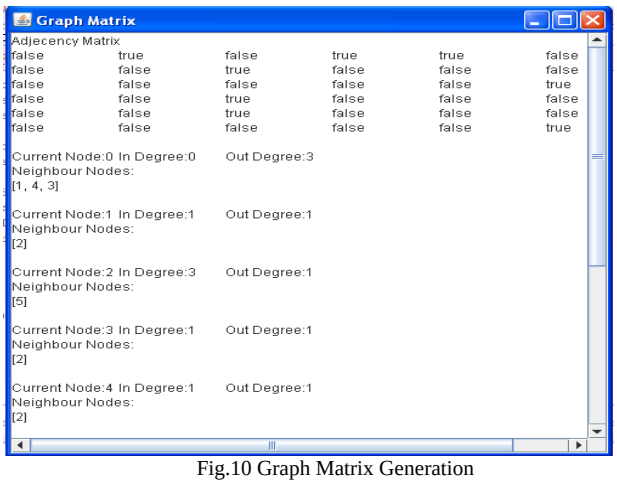

Fig.10 Graph Matrix Generation

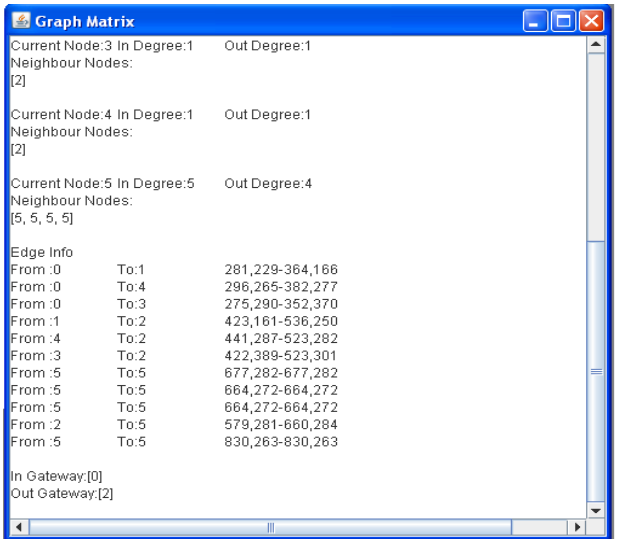

Fig. 11 Find Gateways

RFID readers generate massive datasets, so data must be compressed by eliminating redundancy. Generate a movement graph with n locations and n gateways (items that go from a single initial location to a many ending location and that occur within a certain time interval).we propose an approach that will help in extracting and identifying data patterns at different abstraction levels. In this, the experimental results show how identification of gateways helps user to optimize the query with minimum cost. These results give the information how variation of time takes place when data flows with gateways and without gateways.

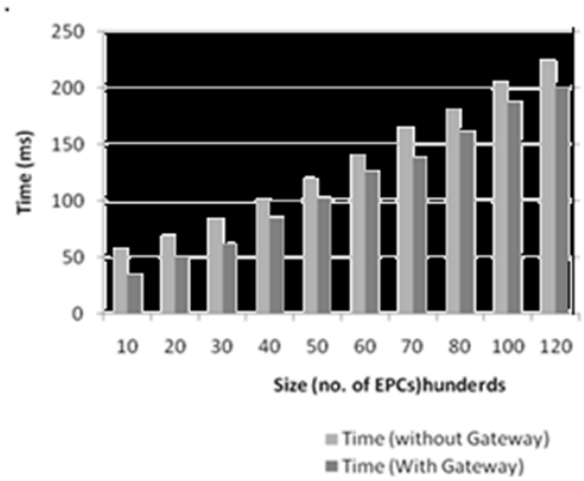

Fig. 12 Clearly Shows the Efficiency with Gateways and without Gateways

XII. CONCLUSION

Proposed model captures the essential semantics of supply chain application as well as many other RFID applications that explore object movements of similar nature. It provides a clean and concise representation of large RFID data sets. Moreover, it sets up a solid foundation for modelling RFID data and facilitates efficient and effective RFID data compression, data cleaning, multidimensional data aggregation, query processing, and data mining.

REFERENCES

[1] Y. Bai, F. Wang, P. Liu, C. Zaniolo, and S. Liu, "RFID DataProcessing with a Data Stream Query Language," Proc. 2007 Int'lConf. Data Eng. (ICDE '07), pp. 1184-1193, Apr. 2007.

[2] S.R. Jeffery, M. Garofalakis, and M.J. Franklin, "Adaptive Cleaning for RFID Data Streams," Proc. 2006 Int'l Conf. Very Large

[3] C. Lee and C. Chung, "Efficient Storage Scheme and Query Processing for Supply Chain Management Using RFID," Proc.2008 ACM Special Interest Group on Management of Data Int'l Conf. Management of Data (SIGMOD '08), pp. 291-302, June 2008.

[4] H. Gonzalez, J. Han, X. Li, and D. Klabjan, "Warehousing and Analysis of Massive RFID Data Sets," Proc. 2006 Int'l Conf. Data Eng. (ICDE '06), Apr. 2006.

[5] S. Agarwal, R. Agrawal, P.M. Deshpande, A. Gupta, J.F. Naughton, R. Ramakrishnan, and S. Sarawagi, "On the Computation of Multidimensional Aggregates," Proc. 1996 Int'l Conf. Very Large Data Bases (VLDB '96), pp. 506-521, Sept. 1996.

[6] Q. Chen, Z. Li, and H. Liu, "Optimizing Complex Event Processing over RFID Data Streams," Proc. 2008 Int'l Conf. Data Eng. (ICDE '08), pp. 1442-1444, Apr. 2008.

[7] C. Floerkemeier and M. Lampe, "Issues with RFID Usage in Ubiquitous Computing Applications," Pervasive computing (PERVASIVE) Lecture Notes in Compute Science, Am. Math. Soc., 2006.

[8] H. Gonzalez, J. Han, and X. Li, "Flowcube: Constructuing RFID Flowcubes for Multi-Dimensional Analysis of Commodity Flows," Proc. 2006 Int'l Conf. Very Large Data Bases (VLDB '06), Sept. 2006.

[9] R.K. Chung, Spectral Graph Theory, vol. 92. Am. Math. Soc. 1997.

[10] EPCIS standard v. 1.0.1, Standard, EPCglobal, http://www.epcglobalinc.org/standards/epcis, 2008.