RFID Data Management

Kamlesh Laddhad
05329014
Guide : Prof. Bernard Menezes
KReSIT, IIT-Bombay

July 17, 2006

Abstract

Radio Frequency IDentification (RFID) is emerging as an all pervasive ubiquitous technology which holds the promise of revolutionizing supply chain management applications. The volume of information generated by such system is enormous. Managing such high volume of data poses the challenges to applications as well as back-end databases. This data is redundant, needs to be consolidated and reduced/transformed so that it occupy less space in the database. In doing so, care must be taken that no useful information is lost. Researchers in database community have presented techniques and models for warehousing as well as cleaning/filtering RFID data. In this report we discuss the RFID technology, characteristics of RFID data, various challenges in handling RFID data and present literature survey of various data warehousing techniques. However this auto-ID[1] technology brings new kinds of risks and security issues which we are not touching as far as this report is concerned.

1 Introduction

Radio Frequency Identification (RFID) is emerging as key technology for a wide-range of applications, including supply chain, retail store, and asset management. RFID offers a possible alternative to barcode identification system. RFID components mainly involve tags and readers(sensors). An RFID tag is a small, low-cost device that can hold a limited amount of data and report that data when queried by a reader. Reader is a device that can recognize the presence of RFID tags and read the information stored in them. RFID uses radio-frequency waves to transfer data between readers and movable tagged objects, thus it is possible to create a physically linked world in which every object is numbered, identified, cataloged. These inexpensive tags can be associated with objects, such as pallets, cases, and even individual items. By placing RFID tag readers at various locations, one can track the movement of objects through supply chain networks, namely, from manufacturers to retailers, then to consumers. Such item-level tracking can greatly enhance the efficiency of business operations. The technology has gained significant momentum in the past few

[1]RFID is invention of Auto-ID center at MIT

years with several high-profile adoptions (e.g. Wal-Mart, Albertson, U.S. Department of Defense, British airport, etc). The U.S. Department of Defense uses RFID to manage shipments to armed forces worldwide. Several major retail chains, including Wal-Mart, Target, and Albertsons, have asked all their suppliers to tag products at the pallet and case level.

Nevertheless, there are some significant challenges that must be overcome before these benefits are realized. There are lots of technical issues raised by RFID. By enlarge, these problems are categorized in three categories and RFID technology is being researched in these streams.

1. *The physics of building tags and readers:* RFID tags has extremely few gates, and many of these are taken up by logic required for basic operation. So very little is left for computing. Also, radio-frequency has some issues with operation under certain physical mediums.

2. *The privacy and safety issues:*

   (a) Because of very low computing power, symmetric encryption schemes such as AES, hash functions such as SHA1, or pseudo-random functions are not possible on today's RFID tags.

   (b) RFID tags communicate with the reader by passively modulating a radio signal broadcasted by the reader. Because a reader is little more than a radio transceiver, attackers are able to obtain illegitimate readers that can be used to query RFID tags from some distance. Counterfeit using of fake tags is also possible.

   (c) Because the communication between reader and tag is wireless, there is a possibility for third parties to eavesdrop on these signals.

3. *The software architecture required to collect, filter, organize, and answer online queries*: Unlike bar-codes, where information is scanned only when someone passes a printed label in front of a reader, RFID scanning is always on. Reader keeps reading tags and observations keeps piling in the database. Full-fledged implementation of RFID with item-level tagging is predicted to generate around 7 terabytes of data every day at Wal-Mart [2].

This report lays stress only on the third stream of research namely data management issue. This stream is quite young and very few techniques have been proposed. EPC-IS[Har03] and PML Core [FAOH03] are the RFID system standardization efforts by auto-ID center. [Har03] summarizes the data characteristics, models data as events and provide some reference relation to represent data. Dynamics Relationship ER (DRER) [WL05] is an expressive temporal data model which enables support for baic queries for tracking and monitoring RFID tagged objects. A simple observation that objects move together in initial stages led to some new proposals. Hu et al. [HSCS05] uses bitmap datatype to compress the infomrmation corresponding to objects that move together.

[2] Venture Development Corporation[vdc], a research firm, reported in a survey.

RFID-Cuboids [GHLK06] are a new warehousing model that preserves object transitions while providing significant compression and path-dependent aggregates. FlowCube [GHL06] is method to construct a warehouse of commodity flows. Some of these are simple representation of various relationship in the Relational DBMS (e.g. [WL05])

The rest of the report is organized as follows. Section 2 quotes different characteristics of RFID data. Section 3 explains challenges in data management. Section 4 presents a literature survey of different models proposed for handling RFID data. In Section 5 we compare between different approaches, enlists advantages and we critique proposed models. Finally section 6 sets the stage for coming stages for thesis.

2 RFID Data Characteristics

RFID systems can be used in diverse applications but the fundamental characteristics for the data remains the same.

1. *Temporal and dynamic*: Applications dynamically generate observation(readings). Objects location as well as containment relationship among objects change along the time as a result, RFID data carry state changes.

2. *Inaccuracy of data*: Sometimes non-existing tag may be incorrectly read (False positive reads) or reader may miss tags which were in its vicinity (False negatives). Also reader may read a same tag more than once. Certainly such erroneous data has to be semantically filtered.

3. *Continuous Streaming*: Number of RF tags are proportional to number of items being serviced/tracked and number of readers are proportional to traceable strategic locations/areas. In typical scenarios, tagged object stay in place for longer duration and readers records their existence on continuous basis in periodic intervals. A tuple is inserted into database each time a tag is read by reader. Such simple observations keeps piling in database and produce redundancy. This continuousl streaming data must be filtered. Also some kind of compression is needed to make reduce data without loss of information.

4. *Granularity*: The level of granularity for data collection needs to be determined. This factor depends on applications for which RFID system is being implemented. E.g. if the system is deployed at retail store, the granularity of data collection may be an item in the shelf or store. On the contrary, if system is deployed at airport, the granularity will be unit of luggage/baggage.

Summing up all these characteristics, one can observe that we need efficient methods for collecting data, cleaning it, shipping it, installing it at the warehouse and updating derived data.

3 Data Management Challenges

As, mentioned in Section 1, RFID readers are continuously reading the data on tags at periodic instances and data keeps generating. A feature that distinguishes an RFID infrastructure from a normal warehousing infrastructure is the much greater emphasis on station-local activities in the former. In a typical data warehousing setup (e.g., for a department store), it is uncommon for the data to be queried at its source. The emphasis is on aggregating data at a central warehouse, where it can be indexed and queried using grouping and aggregation. In contrast, data generated by the tag readers at a dock door in a distribution center is more likely to be used within the distribution center than away from it.

Motivating example: Suppose a retailer with 3,000 stores sells 10,000 items a day per store. Assume that we record each item movement with a tuple of the form: `(object_epc, location, time)`, where EPC is an Electronic Product Code which uniquely identifies each item. Even if each item leaves only 10 traces before leaving the store by going through different locations, this application will generate at least 300 million tuples per day.
Typical queries on the duration of paths like "what is the average time of product P on the shelf?", or "What is the average time that it took product Q to move from the warehouse to the shelf and finally to the checkout counter in January of 2006?". Answering such flexible high-level queries with such an enormous amount of low-level data pose great challenges to traditional relational data warehouse technologies since the processing may involve retrieval and reasoning over a large number of inter-related tuples through different stages of object movement. No matter to what level data is clustered, indexed or sorted; it is difficult to support various kinds of such high-level queries in uniform and efficient way.

Why traditional data cube model would fail: Suppose we view the cleansed RFID data as the *fact table* with dimensions. `(object_epc, location, time_in, time_out : measure)`. The data cube will compute all possible group-bys on this fact table by aggregating records that share the same values at all possible combinations of dimensions. If we use count as measure, we can get for example the number of items that stayed at a given location for a given month. The problem with this form of aggregation is that it does not consider links between the records. For example, if we want to get the number of items of product P that traveled from the distribution center in location L to stores in location U, it is hard to get this information. We have the count of product P for each location but we do not know how many of those items went from the first location to the second. We need a more powerful model capable of aggregating data while preserving its path-like structure.

4 Literature Survey

There have been many data models proposed for handling massive RFID data. We present a short literature survey about the same. We organize our survey chronologically. The progression toward increasing complexity of models over time seems to be natural, as news systems are designed for increasingly complex and changeable environments.

4.1 The beginning

In 2003, auto-ID researcher M. Herrison presented an EPC-Information service[Har03] which marked the beginning of providing interface for access and persistent storage of EPCs. He modeled data as events which eased maintenance of state history. But his model was not effective on supporting complex queries related to monitoring and tracking RFID objects. This model proposed decomposition of certain complex queries into simple queries in which a query "Where are the 5 CDs that were supposed to be in the last order?" was required to be decomposed in 8 simple queries for execution. Certainly, we needed a better technique to model data. Something which supports RFID objects monitoring and tracking queries fairly easily.

4.2 Temporal ER Model

There has been interesting work on ER based temporal modeling of Information systems at conceptual levels. The main difference between standard ER model and temporal ER models is that the data changes very fast and has dynamic nature in the later. In ER model, all entities and relationships are static or current. Whereas in an typical RFID system, entities are static, but all the relationships are dynamic. That is why temporal model perfectly suits the RFID systems. Very basic temporal model for handling RFID system was proposed by Siemens's Wang et al. as Dynamic Relationship ER (DRER) [WL05]. This was temporal extension of ER model by simply adding a new relationship called 'dynamic relationship'. There were two types of such relationship.
(1) Relationship that generates events: A timestamp is associated with these relationship to represent the occurring time of the event. These relationships are represented by doted lines in Figure 1. These are refferd as even-based relationships.
(2) Relationship that generates state history: Two timestamp attributes *tstart* and *tend* are associated with such relationships. These together capture the lifespan of relationship. These are represented by dashed lines in Figure 1.

In typical scenario, based on this model we have following schemas:
Static entities: SENSOR, OBJECT, LOCATION, and TRANSACTION.
State-based dynamic relationships: SENSORLOCATION generated from SENSOR and LOCATION, OBJECTLOCATION generated from OBJECT and LOCATION, CONTAINMENT generated from OBJECTand itself
Event-based dynamic relationships: OBSERVATION generated from OBJECT and SENSOR, and TRANSACTIONITEM generated from OBJECT and TRANSACTION.

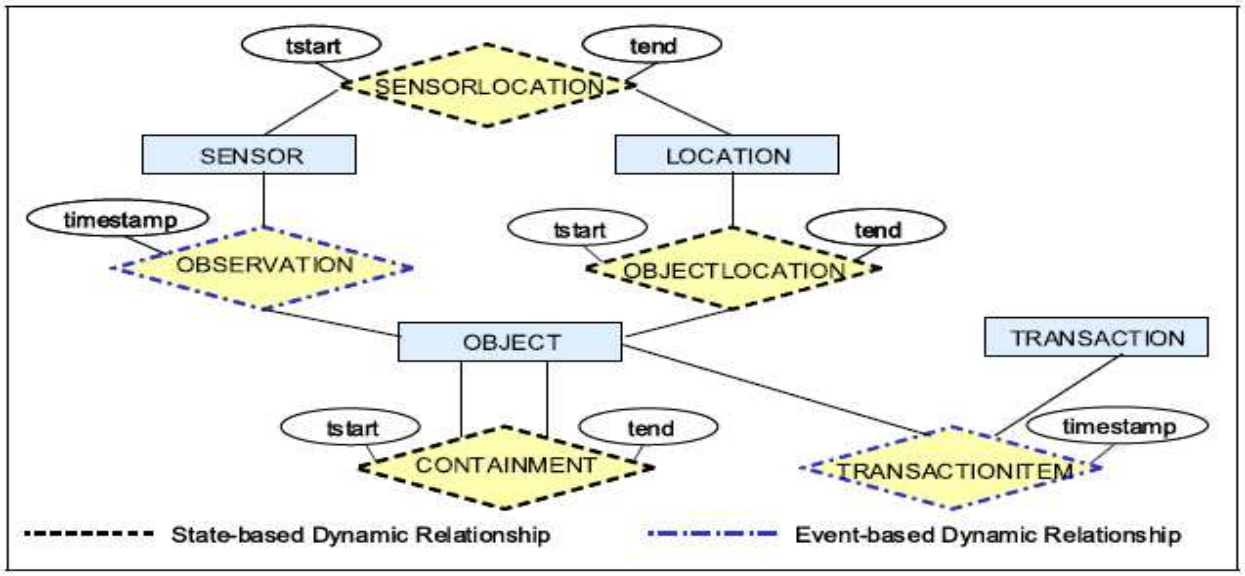

Figure 1: Example ER Model

It is straightforward to implement DRER model in relational database system[1]. There are two types of mappings from the data model to tables. Entities are mapped directly as entity tables. For a state-based dynamic relationship between two entities, it is mapped as a table consisting of keys from both entities, plus an interval [tstart, tend] to represent the lifespan in which the relationship or the state exists. For an event-based dynamic relationship, it is mapped as a table consisting of keys from both entities, plus a timestamp to represent the time when the event occurs. Normal (static) relationships, if any, can be mapped as tables according to common ER mapping rules.

We demonstrate some of the standard tracking and monitoring example queries over this model.

1. **Missing RFID Object Detection**:

   (a) Find when and where object holding EPC= 'MEPC' was lost.

```
SELECT location_id, tstart, tend  FROM OBJECTLOCATION
WHERE epc='MEPC' and tstart = (
   SELECT MAX(o.tstart) FROM OBJECTLOCATION o
   WHERE o.epc='MEPC'
)
```

   (b) Check if there are missing objects at current location C, knowing that all objects were complete at previous location L at time T.

```
SELECT l.epc FROM OBJECTLOCATION l WHERE l.location_id = 'L'
  AND l.tstart <= 'T' and l.tend >= 'T' AND l.epc NOT IN (
    SELECT c.epc FROM OBJECTLOCATION c WHERE c.location_id = 'C'
```

[1]The example is self explanatory

)

2. **RFID Object Moving Time Inquiry**: How long did it take to supply object 'OEPC' from location S to location E?

```
SELECT (e.tstart-s.tstart) AS supplying_time FROM OBJECTLOCATION e, OBJECTLOCATION s
   WHERE e.epc = 'OEPC' AND s.epc='OEPC' AND s.location_id ='S' AND e.locaiton_id='E'
```

This states that once DRER modeling is done, the queries for monitoring and tracking the RFID tagged objects just become standard SQL on these static and dynamic relationship tables and can be mapped in any of the existing database systems. However this model does not at all talk about the system load and problems associated with continuously streaming data. Also does not consider any compression mechanism as explained section 2.

A key observation that though individual item needs to be tracked, usually items move together in large groups through early stages and they can be efficiently tracked by tracking the group to which they belong. Also, data analysis occurs at higher abstraction levels. This observation led to couple of data modeling techniques which uses compression techniques to save space and computation time by storing observations about group which moves together rather than individual objects. We explain both these techniques in greater details in following sections. Section 4.3 explains the RFID-Cuboid architecture and 4.4 explains the use of bitmap datatype in compression.

4.3 RFID-Cuboids

We observed in section 3 that traditional data warehousing techniques does not have links between different tuples in the relation. To come up with this limitation Gonzalez et al. proposed RFID warehouse architecture [GHLK06] that contains a fact table, **stay** composed of cleansed RFID records, an information table, **info**, that stores path-independent information for each item that is constant regardless of the location of the item such as manufacturer, lot number, etc., and a **map** table that links together different records in the fact table that form a path. The stay, info, and map tables aggregated at a given abstraction level are refereed as RFID-Cuboid. Map table links records from the stay table in order to preserve the original structure of the data.
This model takes advantage of the fact that individual items tend to move and stay together (especially at higher abstraction levels) and collapse multiple object movements into a single record without loss of information.

Bulky object movements: Since a large number of items travel and stay together through several stages, it is important to represent such a collective movement by a single record no matter how many items were originally collected. As an example, if 1,000 boxes of milk stayed in location $\texttt{loc}_A$ between time $\texttt{t}_1$ and $\texttt{t}_2$, it would be advantageous if only one record is registered in the database rather than 1,000 individual RFID records. The record would have the form: (gid, prod, $\texttt{loc}_A$, $\texttt{t}_1$, $\texttt{t}_2$, 1000), where 1,000 is the count, prod is the product id, and gid is a generalized id which will not point to the 1,000 original EPCs but instead point to the set of new

gids which the current set of objects move to. For example, if this current set of objects were split into 10 partitions, each moving to one distinct location, gid will point to 10 distinct new gids, each representing a record. The process iterates until the end of the object movement where the concrete EPCs will be registered. By doing so, no information is lost but the number of records to store such information is substantially reduced.

1. **Information Table**: The information table stores path-independent dimensions such as product name, manufacturer, product price, product category, etc. Each dimension can have an associated concept hierarchy. All traditional OLAP operations can be performed on these dimensions. For example, one could drill-down on the product category dimension from 'clothing' to 'shirts' and retrieve shipment information only on 'shirts'.
   Each entry in Info is a record of the form `( (EPC_list); (`d_1`, ... , `d_m`) : (`m_1`, ... , `m_i`) )`, where the `EPC_list` contains a set of items that share the same values for dimensions d_1, ... , d_m, and m_1, ... , m_i are measures of the given items, e.g., price.

2. **Stay Table**: Each entry in stay is a record of the form `(gids; location; time_in; time_out) : (`m_1`, ... , `m_k`) )`, where `gids` is a set of generalized record ids each pointing to a list of RFID tags or lower level gids, `time_in` and `time_out` are respective times when the items entered and left the `location`. If the items did not leave the location, `time_out` is `NULL`. m_1, .. ,m_n are the measures recorded for the stay, e.g., count, average time at location, and the maximal time at location.

3. **Map Table**: There are couple of advantages in using map table instead of recording the complete EPC lists at each stage:

   (a) Data compression:. If we assume that 10000 items move in the system in groups of 10000, 1000, 100, and 10 through 4 stages, instead of using 40,000 units of storage for the EPCs in the stay records, we use only 1,111 units(1000 for the last stage, 100, 10, and 1 for the ones before). This is significant reduction in storage space.

   (b) Query processing efficiency: Suppose each map entry were given a path-dependent label. To compute, for example, the average duration for milk to move from the distribution center (D), to the store backroom (B), and finally to the shelf (S), we need to locate the stay records for milk at each stage. To get three sets of records D, B, and S, one has to intersect the EPC lists of the records in D with those in B and S to get the paths. By using the map, the EPC lists can be orders of magnitude shorter and thus reduce IO costs.

   The map table contains mappings from higher level gids to lower level ones or EPCs. Each entry in map is a record of the form: (gid; (gid_1, ..., gid_n)), meaning that, gid is composed of all the EPCs pointed to by gid_1, ... ,gid_n. The lowest level gids will point directly to individual items.

So we see that [GHLK06] presents a model that allows high-level analysis to be performed efficiently and extensively in multidimensional space. The model is composed of a hierarchy of highly compact summaries (RFID-Cuboids) of the RFID data aggregated at different abstraction levels. However, their analysis is completely based on assumption that RFID objects tend to move together in bulky mode. but there are number of application where this assumption may not hold.

4.4 Use of Bitmap Datatype

RFID Items can be efficiently tracked by tracking the groups to which an item belongs namely the group of items in the same proximity (e.g. on a shelf, on a shipment) of the group of items with same property (e.g. items of a single product). So Hu, et al. proposed use of bitmap type [HSCS05] for modeling a collection of EPCs that that move together. They introduced such a datatype to compactly represent a collection of identifiers and present set of bitmap access and manipulation routines. The most important benefit of bitmap datatype is loss-less transformation. Also, bitmap datatype helps to improve upon efficiency of typical operations like membership testing, comparison operations, insertion and deletion. Table 1 shows how their technique compactly represents the lists of EPCs using bitmap and does significant storage saving.

An EPC identification scheme uniquely identifies an item and a typical EPC number contains (

s_1	p_1	t_1	epc_{11}, epc_{12}, epc_{13}, ...	s_1	p_1	t_1	$bmap_1$
s_2	p_2	t_2	epc_{21} epc_{22}, epc_{23}, ...	s_2	p_2	t_2	$bmap_2$
...	...	...	...	...	...	...	...

Table 1: Product Inventories with EPC Collections and EPC Bitmaps.

`Header, Manager No., Object Class, Serial No.)`. Header identifies the length, type, structure, version and generation of EPC. Manager number identifies an organizational entity. Object class identifies a class, or type of object to which the tag is attached. Serial number specific instance of the Object Class being tagged. Header, Manager No, and object class are collectively called as epc_prefix and the serial number which is used to identify unique items inside the group is referred as epc_suffix. Instead of storing individual tuples for each item, they compress tuples having same epc_prefix into single record, store the staring(suff_start) and ending(suff_end) sufices and keep a bitmap representing existence of corresponding item in suff_start to suff_end range. See Figure 2.

But bitmap datatype is more appropriate for scenarios requiring initial bulk-load followed by

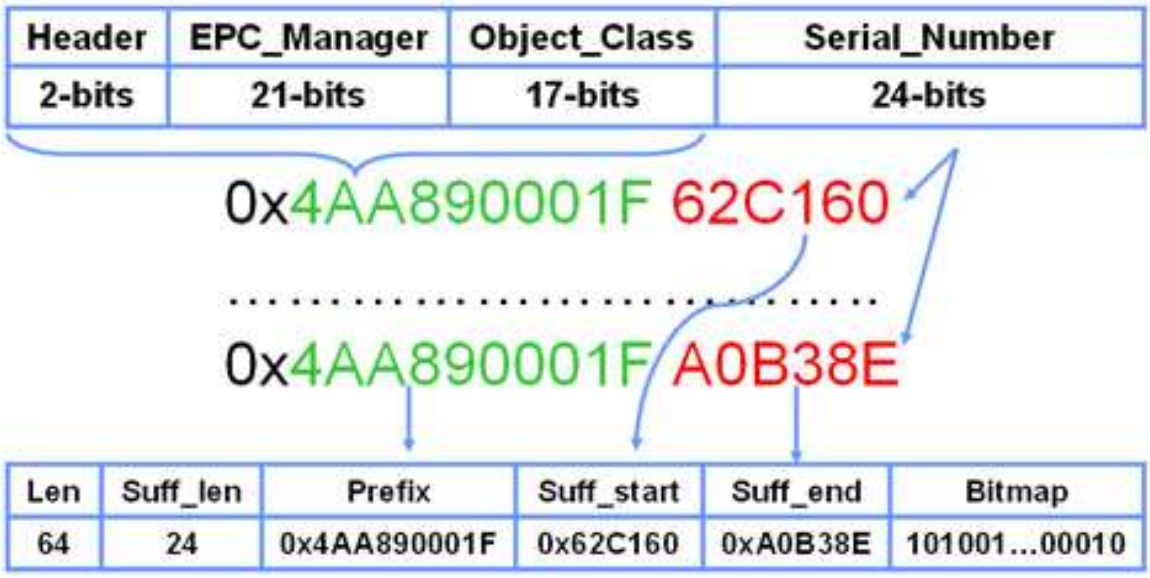

Figure 2: EPC Bitmap Illustration

batch updates at periodic update intervals. But some applications need frequent incremental maintenance. For which they suggest variation of this basic scheme called **'hybrid scheme'** which maintains EPC bitmap in conjecture with a traditional item-level tracking table. In this scheme the inventories are maintained at periodic checkpoints using bitmaps. For changes occurring between checkpoints, an item-level table is maintained. Queries for specific time period are answered by merging latest checkpoint bitmap with appropriate item-level data.

Operations on bitmap:

1. Conversion Operations such as `epc2Bmap` and `bmap2Epc` are useful during storage and extraction of set of EPCs into and from database. The `bmap2Count` operation can be used to count the number of items present in an EPC collection.

2. The logical operations such as `bmapAnd, bmapOr, bmapMinus, bmapXor` helps in determining changes such as what items are removed/added to collection between a particular interval of time(t_1 to t_2), what items were present at t_1 and t_2, etc.

3. Maintenance operation such as `bmapInsert` and `bmapDelete` inserts/deletes a new item to and from given collection.

4. Existence operation `bmapExist` detects the presence/absence of specific item in the collection.

5. Comparison operation `bmapEqual` compares two collections for equality.

Example Query using some of the functions:

1. Following query determines the items added to a given shelf between time t1 and t2.

```
SELECT bmap2Epc(bmapMinus(s2.item_bmap, s1.item_bmap))
```

```
FROM Shelf_Inventory s1, Shelf_Inventory s2
WHERE s1.shelf_id = <sid1> AND s1.shelf_id = s2.shelf_id
    AND s1.time = <t1> AND s2.time = <t2>;
```

2. Lets say an additional table is maintained which keeps track of a property pertaining to book collections in a book store so following query determines the shelves where the books with property 'Adventure' and 'Romance', are currently present in the library.

```
SELECT s.shelf_id FROM Shelf_Inventory s WHERE bmap2Count(
   bmapAnd( s.item_bmap, SELECT bmapAnd(p.Adventure, p.Romance) FROM Propery_Inventory p)
) > 0; AND s.time=<current_date>;
```

5 Summary

We observe that research in this field moves through variety of approaches. Hrrision of auto-ID center started worked for standardization of EPS Information Service [Har03]. He broke the ice with very basic technique to model the RFID data as events. Though the model was not very efficient on complex queries, it started the third research stream for RFID technology. Dynamic relationship ER model [WL05] was the follow up work in ER based temporal modeling of information systems. This model divided various interactions between RFID entities as event based and state based relationships and presented an extension to traditional ER models using such dynamic relationships. This model supported complex queries for tracking and monitoring RFID objects but did not consider erroneous nature of RFID data. This model did not talk about compressing data to remove redundancy.

[HSCS05] and [GHLK06] explored the fact that tagged objects move together in initial stages and analysis occurs at higher abstraction levels. So there is a redundancy in data and a chance to compress tuples corresponding to objects that move together in groups. Hu et al. presented bitmap datatype [HSCS05] for this purpose. This approach is easier to implement in existing Database systems. But it is particularly appropriate for scenarios requiring initial bulk-load followed by batch updates at periodic intervals. For applications requiring frequent incremental maintenance, they propose a hybrid variation of the inventory tracking scheme. RFID-Cuboids [GHLK06] are a new warehousing model to preserve object transitions by keeping a map table which connects tuples in fact table and maintains hierarchial structure of object movement. Both these method do not talk about erroneous nature of data and need to clean and filter the continuously streaming data.

6 Future Plans

[HSCS05] represents the EPC collection using bitmap datatype for object that move in group. This approach maintains one record for all items having same epc_prefix. For all items having epc_suffix different are represented by bitmap structure(explained in section 4.4). This method reduces storage space if the epc_suffixes in the collection are contiguous because in that case, bitmap

is a well formulated string. But in many cases, items in a collection may not have contiguous EPCs. So the bitmap generated will be sparse having lot of zero bits. There is a possibility that we use some standard encoding technique like Huffman coding, etc and compress this bitmap in order to save more storage. This will involve encoding and decoding operations every time we access a bitmap. This may affect efficiency of bitmap operation while doing incremental updates. So, this technique will also be suitable for application scenarios where we require bulk loading and batch updates. We would analyze and explore this extension in coming stage.

Efficient methods for a multitude of data mining problems for the RFID data such as trend analysis, outlier detection, path clustering are some of the open problems and should be a promising line of future research. We plan to explore this field in more details and apply data mining principles to RFID data in the future stages.

References

[CKRS04] Sudarshan S. Chawathe, Venkat Krishnamurthy, Sridhar Ramachandran, and Sanjay E. Sarma. Managing rfid data. In *Proceedings of the 30th International Conference on Very Large Data Bases, Toronto, Canada*, 2004.

[FAOH03] C. Floerkemeier, D. Anarkat, T. Osinski, and M. Harrison. PML core specification 1.0. auto-id center recommendation. Technical report, Auto-ID Center, Sept 2003.

[GB] Bill Glover and Himanshu Bhatt. *RFID Essentials - Theory in Practice.*

[GHL06] Hector Gonzalez, Jiawei Han, and Xiaolei Li. Flowcube: Constructuing RFID flowcubes for multi-dimensional analysis of commodity flows. In *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea*, 2006.

[GHLK06] Hector Gonzalez, Jiawei Han, Xiaolei Li, and Diego Klabjan. Warehousing and analyzing massive RFID data sets. In *The 22nd International Conference on Data Engineering, Atlanta, GA*, 2006.

[Har03] M. Harrison. EPC information service - data model and queries. Technical report, Auto-ID Center, October 2003.

[HSCS05] Ying Hu, Seema Sundara, Timothy Chorma, and Jagannathan Srinivasan. Supporting RFID-based item tracking applications in oracle DBMS using a bitmap datatype. In *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway*, 2005.

[JAF+05] Shawn Jeffery, Gustavo Alonso, Michael J. Franklin, Wei Hong, and Jennifer Widom. A pipelined framework for online cleaning of sensor data streams. Technical report, EECS Department, University of California, Berkeley, 2005.

[JAF+06] Shawn R. Jeffery, Gustavo Alonso, Michael J. Franklin, Wei Hong, and Jennifer Widom. Declarative support for sensor data cleaning. In *Pervasive*, 2006.

[JGF06] Shawn R. Jeffery, Minos Garofalakis, and Michael J. Franklin. Adaptive cleaning for RFID data streams. In *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea*, 2006.

[rfia] RFID journal. Technical report, http://www.rfidjournal.com/.

[rfib] RFID weblog. Technical report, http://www.rfid-weblog.com/.

[vdc] Venture development corporation(vdc).

[WL05] Fusheng Wang and Peiya Liu. Temporal management of RFID data. In *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway*, 2005.

[WLLB06] Fusheng Wang, Shaorong Liu, Peiya Liu, and Yijian Bai. Bridging physical and virtual worlds: Complex event processing for RFID data streams. In *10th International Conference on Extending Database Technology. Munich, Germany*, 2006.