Developing RFID Database Models for Analysing Moving Tags in Supply Chain Management

Wilfred Ng

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Hong Kong
wilfred@cse.ust.hk

Abstract. The applications of RFID (Radio Frequency Identification) have become more important and diversified in recent years due to the lower cost of RFID tags and smaller tag sizes. One promising area for applying the technology is in Supply Chain Management (SCM) in which the manufacturers need to analyse product and logistic information in order to get the right quantity of products arriving at the right time to the right locations.
In this paper, we present a holistic framework that supports data querying and analysis of raw datasets obtained from different RFID collection points managed by supply chains. First, the framework provides repair mechanisms to preprocess raw RFID data from readers. Second, we present a database model to capture SCM information at various abstraction levels such as items, time and locations, and then discuss the use of SQL query language to manipulate RFID databases. Finally, we present a graph data model called a Tag Movement Graph (TMG) to capture the moving information of tagged objects.

1 Introduction

RFID (*Radio Frequency Identification*) is a technology that allows a sensor (an *RF reader*) to read, from a distance, and without line of sight, a unique EPC (*Electronic Product Code*) associated with a tag [19, 20, 5, 17]. The applications of RFID have become more important and diversified in recent years due to the lower cost of RFID tags and smaller tag sizes. One promising area for applying the technology is in *Supply Chain Management* (SCM) [15] in which the manufacturers need to analyse product and logistic information in order to get the right quantity of products arriving at the right time to the right locations.

However, the amount of RFID data in SCM is noisy and massive (e.g. Walmart's warehouse data can be up to the size of petabytes in scale [17]). There still lacks of a unifying model that is able to capture information arising from multi-level and multi-dimensional RFID data. Importantly, we need to develop a framework that supports managing, querying and analysing the information obtained from different RFID sources. All these new RFID features and requirements bring new challenges for providing seamless integration of techniques of data cleaning, data modeling and data mining.

In this paper, we present a holistic framework that enables the management of RFID information and facilitates advanced analysis of the information. We establish data models for RFID data obtained from SCM in the framework which supports tackling the following RFID data problems:

– How to clean massive RFID raw data and store the data in an effective database?
– How to support storing RFID information arising from SCM?
– How to support querying RFID information arising from SCM?
– How to discover useful tag movement trails and correlated patterns in different levels and dimensions of logistic information?

Figure 1 shows the blueprint of the framework, which can pre-process, repair, and store the data collected from multi-RFID data streams. Within the framework, we use a relational DBMS to store RFID data and develop a query language which is translatable into SQL expressions. This approach is practical to RFID industrials, since they usually have a relational DBMS as one of the SCM infrastructures. The new query language can also be employed to manipulate the scope of the derived RFID graph called *Tag Movement Graphs* (TMGs). The framework supports discovering the tag movement relationships from the TMG and RFID databases. Our proposed notions of *Tag Movement Trails* (TMTs) and *Logistic Correlated Patterns* (LCPs) take data abstraction and the SCM logistic information, such as location topology, object grouping and temporal hierarchies, into consideration.

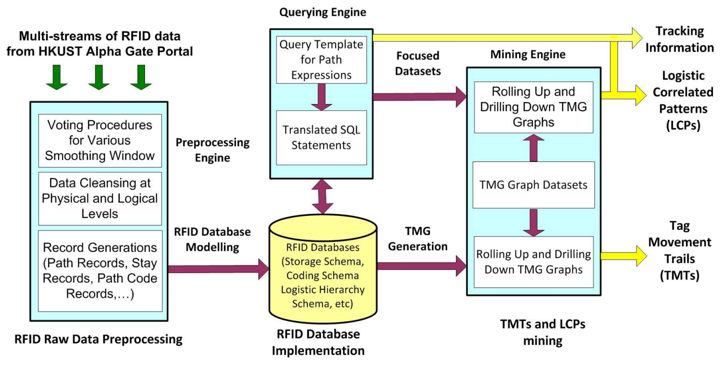
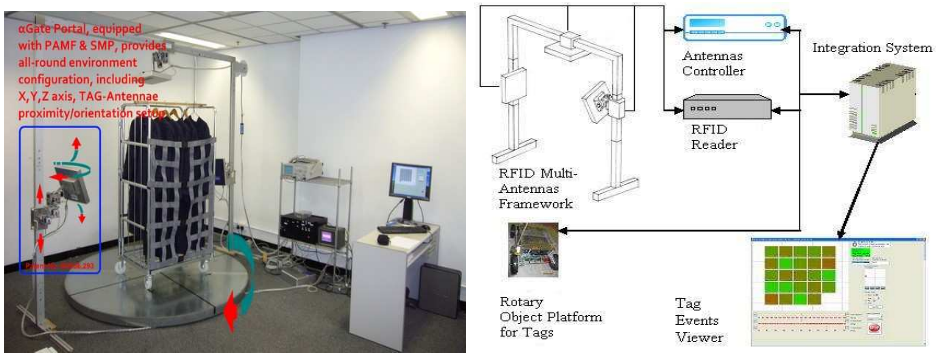

Multi-streams of RFID data from HKUST Alpha Gate Portal
Voting Procedures for Various Smoothing Window
Data Cleansing at Physical and Logical Levels
Record Generations (Path Records, Stay Records, Path Code Records,...)
RFID Raw Data Preprocessing
Preprocessing Engine
RFID Database Modelling
Querying Engine
Query Template for Path Expressions
Translated SQL Statements
RFID Databases (Storage Schema, Coding Schema Logistic Hierarchy Schema, etc)
RFID Database Implementation
Focused Datasets
TMG Generation
Mining Engine
Rolling Up and Drilling Down TMG Graphs
TMG Graph Datasets
Rolling Up and Drilling Down TMG Graphs
TMTs and LCPs mining
Tracking Information
Logistic Correlated Patterns (LCPs)
Tag Movement Trails (TMTs)

Fig. 1. A system view of our proposed framework to support analysing RFID data

The main contributions of this work are related to many interesting modeling and algorithmic issues arising from our proposed framework.

In the modeling aspect, we design the database model as well as the TMG data model to handle the RFID data obtained from multi-stream RFID readers. Figure 2(a) show the setup of the α-gate portal that we used to detect RFID tags from three different readers, thereby generating three RFID data streams that are collected in the detection system in which the connection of the components are presented in Figure 2(b) [21].

In the algorithmic aspect, we develop the coding schemes that support the execution of RFID queries. The algorithm for finding TMTs is an application of the the state-of-the-art research work of graph mining. However, we enrich the techniques to cater for cases of different abstraction levels in TMG data model. The algorithms for finding LCPs are the interesting application of the research discovering correlated patterns in graph databases [14].



(a) (b)

Fig. 2. (a) RFID three-stream (x, y, and z axes) data collection by α-gate portal (b) The setup of the detection system (More details can be consulted from [21])

The rest of the paper is organised as follows. Section 2 presents our cleaning strategies to pre-process RFID raw data obtained from our α-gate portal. Section 3 presents the RFID database model and the coding scheme. We illustrate the application of SQL to support various kinds of RFID queries in Section 4 and present the TMG data model that supports mining of TMTs and LCPs in Section 5. We review the related work of handling RFID data in Section 6. Finally, we give our concluding remarks in Section 7.

2 Preprocessing Multi-Stream Raw RFID Data

In this section, we present the cleaning methods and illustrate the repairing mechanism that preprocesses multi-stream raw RFID Data collected in the α-gate portal and the detection system shown in Figures 2(a) and (b). The main problem in this task related to the RFID raw data preprocessing module presented in Figure 1 is as follows. Given an SCM environment with possibly RFID signal interferences or noises, we devise effective tactics of cleaning multi-RFID data streams obtained from the portal.

In contrast to most existing approaches that focus mainly on low level data [4, 23], we pre-process the data into two phases of physical cleaning and logical cleaning.

Physical cleaning is firstly imposed at the raw data level, in which a smoothing window is employed to remove noises such as *tag jamming* (tag failing to respond to two different reader signals coming at similar time) or *tag blocking* (tag signal is blocked by moisture or metal objects), and logical cleaning is then imposed at the record level to remove multiple, missing or incorrect readings. The challenge is that there is a tension in setting the window size for tracking tags. On the one hand, if we choose a smaller window, we are able to capture the tag movement more accurately. However, it gives rise to more false negatives and undercounting the tags. It is due to the fact that raw data readings can only be picked up by the reader irregularly in the period of tag's presence. On the other hand, if we choose a larger window, we are able to correct more reader's unreliability due to tag jamming or blocking. (c.f. The read rate reported in [12] is roughly 70% of total tag readings in sensor network environments.) However, it gives rise to more false positives and missed tag transitions.

To address the noise problem, we use the α-gate portal shown in Figure 2(a) to develop tactics to clean and repair multi-stream RFID data. We adopt a different approach of "smoothing filter" from SMURF [13], since it is not effective to determine a variable window for all possible RFID data streams. The underlying idea is that we impose a voting system for readings from different window sizes over RFID data streams in order to compensate (i) undercounting tags (missing tag read) and overcounting tags (repeated tag read) and (ii) removing false negatives and false positives.

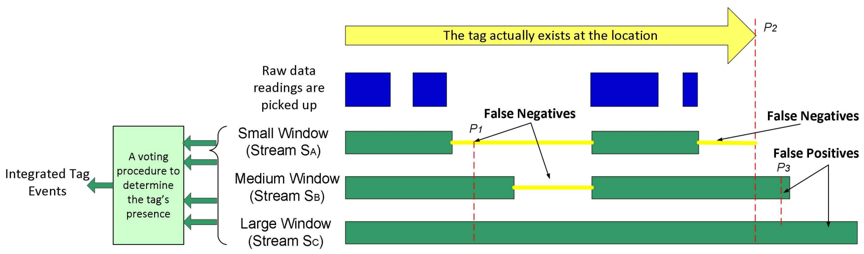
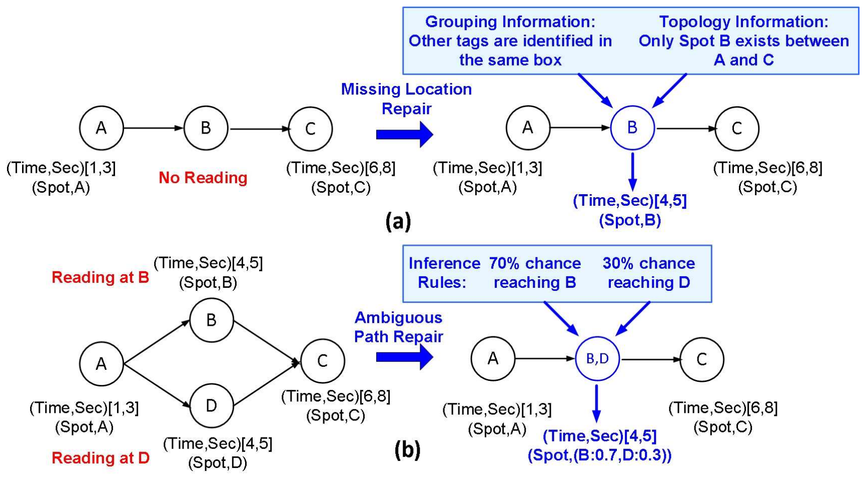

The tag actually exists at the location
P2
Raw data readings are picked up
False Negatives
False Negatives
P1
Small Window (Stream SA)
False Positives
P3
Medium Window (Stream SB)
Large Window (Stream SC)
A voting procedure to determine the tag's presence
Integrated Tag Events

Fig. 3. Using three different smoothing windows for tracking a tag: Stream S_A from a small window avoids most false positives. Stream S_B from a medium window avoids some false positives and some false negatives. Stream S_C from a large window avoids most false negatives. Finally, a voting procedure that obtains "votes" from all streams can effectively remove errors

To illustrate the idea of the voting strategy on the data streams, we show a simplified diagram in Figure 3 that S_A (small window) has a bigger voting weight whenever the conflict of false positives happens (e.g. points P_2 and P_3). However, S_C (large window) has a bigger voting weight whenever the conflict of false negatives happens (e.g. point P_1). S_B (medium window) can be"neutral" such that it carries medium weight on voting. The idea of the voting algorithm is adapted from our earlier work of [16]. We find that the voting strategy is very effective if using more windows of different size that are set on the multi-streams obtained by different antennae configurations (e.g. varying x, y, z and angle settings of the antennas of the α-gate portal in Figure 2).

Logical cleaning following physical cleaning is more interesting. The challenge is how to continue the data cleaning by developing inference rules to repair data (e.g. the EPC is missing in a reader). We classify the logical errors of RFID data into five different classes of (i) missing time - no tracing time data are recorded, (ii) missing location - no location information are recorded, (iii) conflicting path - multiple impossible location location information are recorded, (iv) ambiguous path - multiple possible location information are recorded, and (v) repeated reading - reading are repeatedly recorded.

Due to the space limit, we only highlight two interesting scenarios using three repairing classes in Figure 4, where the pattern $\{Time, Sec\}[t_1, t_2]\{Spot, A\}$ denotes the fact that a tag is detected within a time interval "$[t_1, t_2]$" measured in seconds at spot A.



Grouping Information: Other tags are identified in the same box
Topology Information: Only Spot B exists between A and C
Missing Location Repair
A
B
C
(Time,Sec)[1,3] (Spot,A)
No Reading
(Time,Sec)[6,8] (Spot,C)
(Time,Sec)[4,5] (Spot,B)
(a)
Reading at B
(Time,Sec)[4,5] (Spot,B)
Inference Rules: 70% chance reaching B 30% chance reaching D
Ambiguous Path Repair
D
B,D
Reading at D
(Time,Sec)[4,5] (Spot,D)
(Time,Sec)[4,5] (Spot,(B:0.7,D:0.3))
(b)

Fig. 4. Logical Cleaning: Repair Schemes for (a) Missing location (b) Ambiguous Path

Missing location repair deals with the following scenario shown in Figure 4(a): When a tag goes through a sequence of readers (or locations) A, B and C, B fails to detect the tag. The error may be due to many reasons such as bad reading angles, blocked signals or signal dead zones, which result in weak or no signal readings that are eliminated by the physical cleaning [12, 13]. To repair the error, one rule is to make reference to other location information. For example, by checking the connectivity of the readers, we can deduce that a tag going from A to C must pass B. Another rule is to check the group information. For example, in most SCM settings, tags are moving in groups (e.g. items are contained in boxes or pallets) and thus it is possible to repair some missing tag readings in certain locations if other tags in the same group or a group ID are recorded. Furthermore, *missing time repair* references other tags that go from A to C and then estimates the relative staying time in the detecting region of a reader.

Ambiguous path repair deals with a usual scenario shown in Figure 4(b): A tag goes from A to C passing through B but it is accidentally detected by an abutting reader D placed in another possible path. In this case, we may not be able to decide if the tag passed through B or D. In general, we accommodate the uncertainty in a probability node which represents a possible set of location nodes, each of which is attached a probability as shown in the figure. The assignment of the probability values in the probability nodes is deduced by the rules established by statistical methods that evaluate the sample distribution of the responding tag signal in B and D. An example of such a rule is "if Tag_1 to Tag_{10} passed D, then the probability of Tag_{11} to Tag_{20} going to D is 0.7".

3 RFID Database Model

In this section we develop an RFID database model that supports querying and mining and discuss the implementation issues of database storage.

3.1 RFID Database modeling

We design an RFID database model and implement a scalable storage system for RFID data to support querying and mining. An RFID system consists of three main objects: tags, antennas and readers. A tag T in a location reflects (if T is a passive tag) or emits (if T is an active tag) RF signals within its detection region. When the antenna detects the signal, the reader analyses the signal and stores the EPC and the current timestamp into the database. We represent paths, tags, times and other information in the model and develop path coding schemes for the movement of T in a stored data model. We also incorporate logistic hierarchies and relevant product information into the product data model and the logistic hierarchy data model respectively, which are depicted in Figure 5. This serves as a foundation for the database implementation.

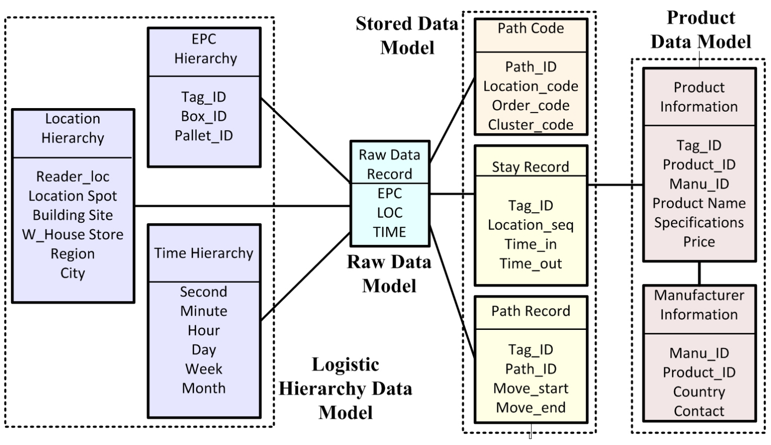


Fig. 5. RFID Database Model with Logistic Hierarchies and Relevant Production Information

To explain the database model in Figure 5, we start by assuming a simple model of RFID raw data in SCM. An RFID reader detects a tag and generates *Raw Data Record* having three attributes (EPC, LOC, TIME) where EPC is a unique ID of a tag and LOC is the location of the (unique) RFID reader and TIME is the time instant of detecting the tag. We then collapse the raw data into the *Stay Record* entity having four attributes (tag_id, loc_code, time_in, time_out) where time_in and time_out are the time instants of the first detection and the final detection of the tag. A stay record represents an RFID tag moving through a location during a specific time interval. However, using a stay record as a stored data model is too simplistic to support the evaluation of path queries that track objects, particularly for those path queries involving many locations, which need to perform self-joining of the table many times. Thus, we develop various sophisticated coding techniques and include *Path Record* and *Path Code* entities into the model.

The basic idea of the coding scheme is that a tag movement path "$L_1 \rightarrow L_2 \rightarrow \cdots \rightarrow L_n$" can be coded by assigning a unique prime number, P_l, to represent all locations (or all readers) and another prime number, P_o, to represent the location order. To generate a unique integer pair (P_l, P_o), we rely on the *Unique Factorization Theorem* for coding locations and the *Chinese Remainder Theorem* for coding their order. However, due to the scarcity of prime numbers, the state-of-the-art method in [15] which supports neither long path coding (e.g. encode a long path of more than 8 locations) nor cyclic path coding (e.g. encode a path in which a tag passed a location twice). The first problem is due to the fact that most programming languages use unsigned integers (32 bits) that only support $2^{32} - 1$, which is less than the product of the first nine prime numbers $2 \times 3 \times 5 \times 7 \times 11 \times 13 \times 17 \times 19 \times 23$. Even for 64 bit unsigned integers (2^{64}-1), it can only support the first 15 prime numbers. In addition, suppose a tag goes through the path "$L_1 \rightarrow L_2 \rightarrow L_3 \rightarrow L_1 \rightarrow L_2 \rightarrow L_3$". In this case, the coding method in [15] fails, since the system of simultaneous *congruences* of Chinese remainder theorem is not applicable here and thus (P_l, P_o) fails to code 2→3→5→2→3→5.

3.2 RFID Data Coding Schemes

To tackle the two problems illustrated in Section 3.1, we need to review some important results from number theory, which is the necessary background for understanding our proposed path coding schemes. Theorem 1 summaries all the relevant results.

Theorem 1. *The following theorems are well-established mathematical results [10].*

1. ***(The Fundamental Theorem of Arithmetic)*** *Any natural number greater than 1 is uniquely expressed by the product of prime numbers.*
2. ***(The Chinese Remainder Theorem)*** *Suppose that $n_1, n_2, \ldots, n_k$ are pairwise relatively prime numbers. Then, there exists x between 0 and $(n_1 \cdot n_2 \cdot n_3 \cdots n_k - 1)$ by solving the following system of simultaneous congruences*

$$x \mod n_i \equiv a_i \; for \; 1 \leq i \leq k.$$

3. ***(Euler Formula for Prime Generation)*** *For every integer x between 0 and 40, $x^2 - x + 41$ is a prime number.*
4. ***(Finite Continued Fraction)*** *Given a finite sequence of positive integers $\langle x_1, x_2, \ldots, x_n \rangle$, there exists a rational number Y given by*

$$Y = \frac{1}{x_1 + \frac{1}{x_2 + \cdots \frac{1}{x_n}}}$$

such that Y uniquely expresses $\langle x_1, x_2, \ldots, x_n \rangle$.

To solve the long path coding problem, we first partition the whole set of locations into different clusters having roughly the same number of locations. Using finite continued fraction in Theorem 1 we are able to represent a cluster_code denoted as C (having a unique positive integer as its id) together with its respective (P_l, P_o). Notably, the clustering can be straightforwardly extended into more than one level within each cluster, which therefore removes the constraint of having no more than 8 prime numbers for coding a path in a cluster.

Suppose there are two clusters coded by two positive integers C_1 and C_2. The subpath in cluster 1 can be computed as loc_code 1 and order_code 1 and similarly notations for the subpath in cluster 2. If a path goes from cluster 1 to cluster 2, we generate the fullpath_code $\mathcal{P}$ as

$$\cfrac{1}{C_1 + \cfrac{1}{loc\_code1 + \cfrac{1}{order\_code1 + \cfrac{1}{C_2 + \cfrac{1}{loc\_code2 + \cfrac{1}{order\_code2}}}}}}.$$

When decoding $\mathcal{P}$, we first check whether it is smaller than 1. If this is the case, then the path covers more than one cluster. We then decompress $\mathcal{P}$ to extract loc_code and order_code in each cluster.

To solve the cyclic path coding problem, we apply Euler's prime number generation formula in Theorem 1. For example, the cyclic path 2→3→5→2→3→5 is coded as "2 $\rightarrow$ 3 $\rightarrow$ 5 $\rightarrow$ 43(x=2) $\rightarrow$ 47(x=3)$\rightarrow$ 61($x = 5$) which can be used to form the system of congruences required by the Chinese remainder theorem of Theorem 1.

We are now ready to present our algorithms to handle long and cyclic paths. For simplicity in presentation, we assume one reader for each location and one level of clustering. We divide the whole set of locations into n clusters, each of which has less than 8 locations. We then code each cluster by an integer and within each cluster a location is coded by a unique prime number n_p. We now represent a path with three integers (loc_code P_l, order_code P_o, cluster_code N_c) only. Loc_code can be computed by using the *Fundamental Theorem of Arithmetic* in Theorem 1, which by definition P_l is the product of all prime numbers associated with the path. Order_code exists according to the *Chinese Remainder Theorem* in Theorem 1 and P_o can be computed by Euclid's algorithm [22]. For example, consider $\{n_1 = 2, n_2 = 3, n_3 = 5\}$ and $(P_o \mod 2) \equiv 1$, $(P_o \mod 3) \equiv 2$, $(P_o \mod 5) \equiv 3$, then P_o can be computed as $((1 \times 3 \times 5 + 2 \times 5 \times 2 + 2 \times 3 \times 3)\text{mod}(2 \times 3 \times 5)) \equiv 23$.

As we are not able to make order_code P_o congruent to the same location twice in the Chinese Remainder Theorem, we need to assign more than one prime number to those repeatedly visiting locations. Here is our proposed solution to this problem. First, we code each location with a prime number as said and this number is not an *Euler Formula Prime*. We call this set of location prime numbers the *Fundamental Location Set* and denote the set by $\mathcal{F}$. Given that the *Stay Records* can be sorted by time_in, if a specific location occurs twice, the first occurrence will be the prime code n from $\mathcal{F}$ and then the second occurrence can be coded by applying Euler Formula by putting $x = n$. The new generated Euler prime numbers do not belong to $\mathcal{F}$. We now ready to present the ideas of path coding in Algorithm 1.

The underlying idea of the decoding process is as follows. First, we decompress the fullpath_code (if it is found to be larger than 1) to identify clusters and for each cluster C decompose loc_code into its corresponding list of all locations p. Second, we know whether there is a cycle in an encoded path by comparing p with $\mathcal{F}$. If there are cycles, after sorting p, we decode those prime numbers that are not in $\mathcal{F}$ by reversing Euler Theorem to get their original set of prime numbers. Finally, the path can be constructed by sorting p by using order_code. We now present the ideas in Algorithm 2.

Algorithm 1 Encoding

```
Input: A fullpath $p$, Fundamental Location Set $\mathcal{F}_i$ for each cluster $c_i$
Output: Fullpath_code $\mathcal{P}$
  Assign a positive integer $n_i$ to each cluster $c_i$
  for each cluster $c_i$ do
    if there are repeated Stay Records in $c_i$ then
      Encode these Repeated locations by Euler Theorem
      Update $\mathcal{E}_i$ by including the Euler's prime numbers
    end if
    loc_code := Product of all prime number in $\mathcal{F}_i \cup \mathcal{E}_i$
    order_code := Output by using $\mathcal{F}_i \cup \mathcal{E}_i$ in the Chinese remainder theorem
  end for
  fullpath_code := Result of a finite continued fraction of $\{n_i$, loc_code, order_code $\}$ for all $c_i$
```

Algorithm 2 Decoding

```
Input: A fullpath_code $\mathcal{P}$, Fundamental Location Set $\mathcal{F}_i$ for each cluster $c_i$
Output: A full path defined by $p$ := $\bigcup p_i$ for all path segment $p_i$ in the clusters
  Decompress $\mathcal{P}$ to identify the ordered set of loc_code and order_code in each cluster
  for each cluster $c_i$ where $i$ preserves the order of the decompressed integer sequence from $\mathcal{P}$
  do
    for all prime numbers $n_p \leq$ loc_code do
      if loc_code % $n_p$ equals to 0 (i.e. the remainder for dividing loc_code by $n_p$ is 0) then
        Add $n_p$ into $p_i$ with an order
      end if
    end for
    for all $n_p$ in $p_i$ do
      Remainder Set $\mathcal{R}$ := $\mathcal{R} \cup \{$order_code % $n_p\}$
    end for
    Sort $p_i$ according to the order in $\mathcal{R}$
    if there are cycles (i.e. $p_i - \mathcal{F}_i \neq \emptyset$) then
      Inverse Euler Theorem to all $n_p$ in $(p_i - \mathcal{F}_i)$
    end if
  end for
```

Example 1. Suppose that a tagged object goes though locations $C \rightarrow B \rightarrow A \rightarrow B \rightarrow C$ in the same cluster. We have $\mathcal{F} = \{1, 2, 3\}$. We use prime numbers to represent these locations as follows: C is denoted as 2, B is denoted as 3, and A is denoted as 5. Then, cluster_code = 1, loc_code = $2 \times 3 \times 5 \times Euler\,(3) \times Euler\,(2) = 2 \times 3 \times 5 \times 47 \times 43 = 60630$, and order_code $= 24773$. To decode the path, we first decompose loc_code into $\mathcal{P} = \{2, 3, 5, 47, 43\}$. Then we can get their order by dividing order_code by all the elements in $\mathcal{P}$ and the remainder set (in order) are $\{1, 2, 3, 4, 5\}$. We sort $\mathcal{P}$ by this order and thus get $2 \rightarrow 3 \rightarrow 5 \rightarrow 47 \rightarrow 43$. As the numbers 47 and 43 do not belong to $\mathcal{F}$, we decode them by reversing *Euler Theorem* to get original prime numbers 2 and 3.

4 RFID Manipulation Languages

In this section we present RFID Manipulation Languages and discuss their SQL processing. The architecture shown in Figure 1 provides a platform to translate different classes of RFID queries into their corresponding SQL statements.

The language for formulating RFID queries is defined by borrowing the notation of XML path expressions. We consider queries and the results can be expressed by the syntax elements such as parent axis (/), ancestor axis (//) and predicate ([]). Let us consider the following RFID raw data and the corresponding stay records in Table 1.

Raw Data Records	(Tag_1,L_1,1), (Tag_1,L_1,2), (Tag_1,L_2,3), (Tag_1,L_2,4), (Tag_1,L_3,5), (Tag_1,L_3,6), (Tag_2,L_1,3), (Tag_2,L_1,6), (Tag_2,L_2,7), (Tag_2,L_2,8), (Tag_2,L_3,9), (Tag_2,L_3,10)
Stay Records	Tag_1: L_1[1, 2] $\rightarrow$ L_2[3, 4] $\rightarrow$ L_3[5, 6] Tag_2: L_1[3, 6] $\rightarrow$ L_2[7, 8] $\rightarrow$ L_3[9, 10]

Table 1. RFID Raw Data and Stay Record

SQL is employed to support three types of queries of *tracking queries*, *path-oriented queries*, *containment queries*.

- Tracking queries aim to obtain the path information for a given tag.
- Path-oriented queries aim to obtain information in a given path expression.
- Containment queries aim to obtain information from the relationships between tags, boxes and pallets.

To support processing of above three classes of queries, we rely on the RFID database model presented in Figure 5 for handling the path and tag information. We illustrate the language by using the following RFID queries and their corresponding SQL translation.

We can formulate tracking queries that require tag locations or location history. The query results are given according to Table 1.

Query Q_1: Find the path for tag_id $= Tag_1$.

Results: $L_1/L_2/L_3$ (decoded from the path code by Algorithm 2).

Tracking queries require tag locations or location history. Q_1 traces Tag_1 and the query can be translated into an SQL statement by assuming the tables corresponding to Figure 5: pathrecord(tag_id, path_id, move_start, move_end), pathcode(path_id, loc_code, order_code, cluster_code) and stayrecord(tag_id, loc_code, time_in, time_out) as the following SQL expression.

```
SELECT decode(loc_code, order_code, cluster_code)
FROM pathrecord P, pathcode C
WHERE tag_id = 'Tag1' AND P.path_id = C.path_id
```

We are able to formulate path oriented queries to obtain information in a given path expression. We assume $n_i = Prime(L_i)$ and the MOD function provided by Oracle DBMS in SQL translation.

Query Q_2: Find the tag_ids that go to L_3 via L_1.

Expression: Expressions (based on XPath convention) $\langle$//L1//L3$\rangle$

Results: Tag_1, Tag_2

```
SELECT tag_id
FROM pathrecord P, pathcode C
WHERE MOD(C.loc_code, $n_1 \times n_3$) = 0 AND
MOD(C.order_code, $n_1$) < MOD(C.order_code, $n_3$) AND P.path_id = C.path_id
```

Query Q_3: Find the tag_ids going from L_1 to L_2 where the duration at L_1 is less than 2.
Expression: $\langle //L_1[(\text{EndTime} - \text{StartTime}) \leq 2]/\ L_2 \rangle$
Results: Tag_1

```
SELECT S.tag_id
FROM pathrecord P, pathcode C, stayrecord S
WHERE MOD(C.loc_code, $n_1 \times n_2$) = 0 AND
MOD(C.order_code, $n_1$) + 1 = MOD(C.order_code, $n_2$) AND P.path_id = C.path_id
AND P.tag_id = S.tag_id AND S.loc_code = $n_1$ AND (S.Time_in - S.Time_out) < 2
```

For containment queries that involves product information and manufacturer details, SQL expressions can be formulated in a similar way as usual data warehouse queries [9], which are well-known SQL work and therefore are not detailed here.

5 Tag Movement Graph (TMG) Model

In this section, we present a graph-theoretic data model to capture RFID tag trail information and discuss how to discover *tag movement trails* (TMTs) and *logistic correlated patterns* (LCP) defined within the model.

5.1 Capture Frequent Tag Movement Trails in a TMG

We establish a graph-theoretic data model to capture RFID tag movement information. This can be achieved by decomposing the RFID database into a TMG via adapting some established "tables to graph" algorithms such as BANKS [1]. Based on the logistic hierarchy model in Figure 5, a node in the TMG graph is annotated with SCM information along three dimensions of the object tags (i.e. with an EPC hierarchy: object $\rightarrow$ box $\rightarrow$ case $\rightarrow$ pallet), locations (i.e. with a location hierarchy: spot $\rightarrow$ site $\rightarrow$ store $\rightarrow$ region) and time (i.e. with a temporal hierarchy: minute $\rightarrow$ hour $\rightarrow$ day $\rightarrow$ week).

For example, a node in a TMG at the lowest level of the mentioned three hierarchies is annotated as $\{$(EPC,object), (Loc,Spot), (Time,Sec)$\}$ to store the object coded as EPC. A higher level annotation on the node can be formulated as $\{$(EPC,Box), (Loc,Store), (Time,Day)$\}$ or a further higher level as $\{$(EPC,Pallet), (Loc,Region), (Time, Month)$\}$. We implement data warehouse operators such as roll up and drill down to control different levels of abstraction in each dimension. In contrast to the RFID Cuboid [9], each node of TMG can be annotated with the data in all the dimensions and each edge with the aggregated information of transitions such as the total number of tag read. We can restrict the TMG on a focused dataset obtained by formulating an RFID manipulation language expression as discussed in Section 4.

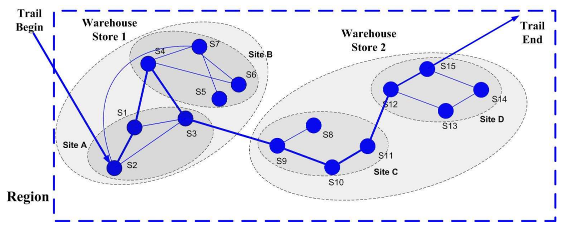


Fig. 6. A TMG Graph viewed at multi-levels of location: Spot $\rightarrow$ Site $\rightarrow$ Store $\rightarrow$ Region

We now show a simplified location view of a TMG in Figure 6. Within the TMG graph model, we discover frequent Tag Movement Trails (TMTs) that meet some threshold criteria in various abstraction levels. We can see that a simple form of a TMT trail can be defined as $(E, S_2 \rightarrow S_1 \rightarrow S_4 \rightarrow S_3 \rightarrow S_9 \rightarrow S_{10} \rightarrow S_{11} \rightarrow S_{12} \rightarrow S_{15}, T_{Spot})$ which represents a trajectory of tag E running in different location spots at different times specified by T_{Spot}. The TMT trail can be rolled up to the site level as $(E, Site_A \rightarrow Site_B \rightarrow Site_A \rightarrow Site_C \rightarrow Site_D, T_{Site})$, or to the warehouse level as $(E, Store_1 \rightarrow Store_2, T_{Store})$. The abstraction for tags E and time T (e.g. T_{Spot} in second, T_{Site} in minute, and T_{Store} in hour) can also be rolled up similarly.

The TMTs can be explored by performing a generalization of some standard graph exploration methods such as DFS (depth-first search) or BFS (breadth-first search) [22]. The parameters that judge various approaches are defined in terms of memory consumption and number of iterations. We define a stopping condition of the exploration as follows: using BFS-like algorithm we construct a tree-like structure with the start node as its root wherein each branch of the tree is explored until its probability falls below a cut-point. A branch of the tree having probability above the cut-point is a candidate trail if it still has some expansions to be evaluated. The process continues until a longest TMT can be found. There is also a trade-off between mining more trails by lowering the cut-point and improving the efficiency by reducing the depth of exploration.

5.2 Logistic Correlated Patterns on TMG

A *Logistic Correlated Pattern* (or simply an LCP) is a nonempty set of correlated items having distinct logistic attributes (time, trails and other product quality parameters), which are both multi-level and quantitative in nature [14]. An example item is "(Time, minute)[100, 200]" representing from 100 to 200 minutes and an example of 2-patterns is "(Time, minute)[100, 200](Trail, spot)[S1, S5]" with the logistic attribute set $\{$(Time, minute), (Trail, spot)$\}$ and the interval set $\{$[100, 200], $[S_1, S_5]\}$.

LCPs reveal the fact that "The time (100 to 200 sec) spent in the trail $S_A \rightarrow S_B$ (coded as 1$\rightarrow$2) is too long and makes the milk spoil" or "The number of tags passing through the trail $S_A \rightarrow S_B \rightarrow S_C$ (coded as 1$\rightarrow$2$\rightarrow$3) is between 1K and 2K in the first day". They can be expressed as LCPs "(Time, minute)[100, 200](Trail, Spot)[1, 2](quality)[spoil]" and "Sum(Object, tag)[1K, 2K](Trail, spot)[1, 3](Time, day)[0, 1]". Mining the RFID database and the TMG graph together give rise to the discovery of rich hidden SCM information for further analysis.

6 Related Work

Initial studies of RFID technologies focused mainly on the issues arising from low level abstraction such as signal filtering and resolution, RFID sensitivity tuning and RFID benchmarking and standardization [8, 2]. As the amount of RFID data becomes extremely large (e.g. Walmart generates RFID data in the petabyte scale each day [17]), the problem of applying database and data mining technologies to handle RFID data is increasingly necessary and important. There are many interesting issues for handling RFID data such as stream processing [24, 5, 13], managing RFID data [23, 9], cleaning raw RFID data and RFID data mining [18, 7]. However, there still lacks of an integrated framework to support more advanced data analysis.

The work related to event processing can be found in Wang et al. [24], which considers temporal RFID events and formalizes the specification and semantics of RFID events and rules. Also, they proposed a method to detect RFID complex events efficiently. Bai et al. [4, 5] explored the limitation of SQL in supporting the temporal event detection and discussed an SQL-based stream query language to provide comprehensive temporal RFID event detection. The system architecture for managing RFID data is also discussed in [3, 6, 11].

An important issue for RFID applications is that the collected raw data has different sorts of errors such as duplicate readings and missing readings. To clean the raw data, SMURF [13] was proposed to control the window size of the smoothing filter adaptively using statistical sampling. [4] also proposed several methods to filter RFID data. However, there is still a lack of work to address the errors in high level abstraction and handling multi-streams of raw RFID data.

In the area of RFID data modeling, Wang et al. [23] proposed the Dynamic Relationship ER Model (DRER) which includes a new relationship (dynamic relationship). They also proposed methods to express temporal queries based on DRER. Gonzalez et al. [9] proposed a new data warehousing model for the object transition and a method to process a path selection query. Lee and Chung [15] proposed a storage scheme to aid processing a set of RFID queries such as tracking and path-oriented queries. The coding schemes apply some important results from the prime number theory. Unlike our coding schemes presented in Algorithms 1 and 2, only very few location nodes can be handled due to the scarcity nature of prime numbers and no cycle is allowed to happen in their scheme. There are few works related to mining RFID [7] but still many issues such as analysing patterns and trails that have not been adequately explored.

7 Concluding Remarks

We present a holistic framework that supports collecting and analysing RFID raw data in a SCM setting. Within the framework, we illustrate the techniques of modeling and storing RFID data and discuss how to make RFID queries translatable into SQL expressions. This approach is practical to RFID industrials, since they usually have relational DBMSs as one of the SCM infrastructures. To discover more interesting SCM information, we also propose the notions of TMTs and LCPs, which take data abstraction and the SCM logistic information, such as location topology, object grouping and logistic hierarchies, into consideration. The proposed framework provides much stronger support to business activities that involve complex movements of goods in large quantities. This work also demonstrates the application of many fundamental research areas such as data warehouse operations and data mining on graphs. Throughout the paper, we have discussed various issues from modeling and system view points. To further demonstrate the feasibility of the framework, we are collaborating with our RFID lab industrial partners (see [20], Partners) to gain user feedback as a future work.

Acknowledgements. This work is partially supported by Hong Kong RGC GRF under project number 617610.

References

1. B. Aditya, et al. Sudarshan: BANKS: Browsing and Keyword Searching in Relational Databases. In Proc. of VLDB, pp. 1083-1086, 2002.
2. R. Angeles, RFID Technologies: Supply-chain Apps. and Implementation Issues, 2005.
3. C. Bornhövd, et al. Integrating Automatic Data Acquisition with Business Processes - Experiences with SAP's Auto-ID Infrastructure. In: Proc. of VLDB, 2004.
4. Y. Bai, F. Wang, and P. Liu. Efficiently Filtering RFID Data Streams. In: Proc. of VLDB Workshop on Clean Databases, 2006.
5. Y. Bai, F. Wang, P. Liu, C. Zaniolo, and S. Liu. RFID Data Processing With a Data Stream Query Language. In: Proc. of ICDE, 2007.
6. S. S. Chawathe, V. Krishnamurthy, S. Ramachandran, and S. Sarma. Managing RFID data. In: Proc. of VLDB 2004.
7. Elio M., "A Framework for Outlier Mining in RFID data", In: Proc. of IDEAS, 2007.
8. EPCGlobal, Inc. http://www.epcglobalinc.org/home.
9. H. Gonzalez, J.i Han, X. Li, D. Klabjan, Warehousing and Analysing Massive RFID Data Sets, In: Proc. of ICDE 2006.
10. Hardy, G. H.; Wright, E. M. , *An Introduction to the Theory of Numbers*, 1979
11. J. E. Hoag and C. W. Thompson. Architecting rfid middleware. IEEE Internet Computing, 10(5): 88-92, 2006.
12. S. R. Jeffery, et al. A Pipelined Framework for Online Cleaning of Sensor Data Streams. In: Proc. of ICDE. 2006.
13. S. R. Jeffery, M. N. Garofalakis, and M. J. Franklin. Adaptive Cleaning for RFID Data Streams. In: Proc. of VLDB, pages 163-174, 2006.
14. Y. Ke, J. Cheng and W. Ng. Correlated Pattern Mining in Quantitative Databases. ACM Transactions on Database Systems, 33(3), 2008.
15. C-H Lee, C-W Chung, Efficient Storage Scheme and Query Processing for Supply Chain Management using RFID, In: Proc. of SIGMOD 2008.
16. W. Ng, L. Deng and D.L. Lee. Spying Out Real User Preferences in Web Searching. ACM Transactions on Internet Technology, 2007.
17. Palmer M, Principles of Effective RFID data management, Enterprise System, March 2004.
18. J. Rao, S. Doraiswamy, H. Thakkar, and L. S. Colby. A Deferred Cleansing Method for RFID Data Analytics. In: Proc. of VLDB, pages 175-186, 2006.
19. http://www.hk-rd.com/, Hong Kong RFID.
20. http://www.rflab.org, HKUST RFID Lab.
21. http://www.cse.ust.hk/News/RFIDAward2008, α-Gate Portal Award News.
22. R. Sedgewick, P. Flajolet, An Introduction to the Analysis of Algorithms.
23. F. Wang and P. Liu. Temporal Management of RFID Data. In: Proc. of VLDB, pages 1128-1139, 2005.
24. F. Wang, S. Liu, P. Liu, and Y. Bai. Bridging physical and virtual worlds: Complex event processing for rfid data streams. In: Proc. of EDBT, 2006.