RFID Data Management, Aggregation and Filtering

Oleksandr Mylyy

Hasso Plattner Institute at the University of Potsdam, Prof.-Dr.-Helmert-Strasse 2-3,
14482 Potsdam, Germany
oleksandr.mylyy@student.hpi.uni-potsdam.de

Abstract. Radio Frequency Identification (RFID) technology enables a new era of business optimization. RFID provides capturing large volumes of data at high velocity and can be used for identifying, locating, tracking and monitoring physical objects without line of sight. However, despite these benefits, this technology poses many new challenges in the current data processing and management systems. RFID data are dynamically changing and time-dependent, RFID observations can contain redundant information, and the information volume generated by such a system can be enormous. Furthermore, low-level implicit RFID observations need to be transformed and aggregated into semantically meaningful data suitable for different enterprise applications. This paper aims to consider a system architecture for managing RFID data based on efficient analyzing, filtering and aggregation techniques for RFID data streams. The usage of these techniques in RFID middleware and by construction of the RFID warehouse is an important topic of our research.

1 Introduction and Motivation

RFID technology gives a possibility to create a physically linked world where every object is numbered, identified, cataloged, and tracked. In contrast to barcodes, RFID does not require line of sight or contact between readers and tagged objects. The main advantages of RFID systems are price efficiency and accuracy of stock management. In addition to emerging applications in retail and distribution, RFID has been gradually adopted and deployed in a wide area of applications, such as aircraft maintenance, baggage handling, laboratory procedures, security and healthcare.

True benefits of RFID technology can only be realized when the tracking information from RFID components is efficiently integrated into business applications. However, there are some significant challenges in the current data processing and management systems that must be overcome. The first of them is the enormous amount of low-level data that must be processed on the fly. RFID observations contain redundant information in form of duplicates, which have to be filtered. Moreover, RFID observations have implicit meanings, which have to be semantically transformed and aggregated. Thus, RFID delivers a granularity that needs to be effectively managed from a quality as well as a quantity perspective. This paper aims to formalize the constructive way for efficient RFID data management, filtering and aggregation.

1.1 Background

Radio Frequency Identification (RFID) systems consist of three components: RFID transceivers or readers (antenna is often integrated into readers), RFID transponders or tags [6], [7], and the application that makes use of the generated data. RFID readers are capable of reading the information stored at RFID tags placed in their vicinity and communicate it through wired or wireless interface to a central database. The electronic product code (EPC) standard [12] defines world wide unique IDs for RFID tags which are stored in their memory and used to uniquely identify each of them. Most RFID applications today use extremely limited in their abilities, inexpensive, passive RFID tags. Such tags do not require batteries and do little more than reveal their stored information using energy of the received signal from RFID reader. The first advantage of this approach is simplicity which makes such tags much cheaper to manufacture and economically feasible to attach them to almost any item in the world. Furthermore, application-independency of these tags enables standards developed for managing RFID data to find a wide cross-industry adoption. In this paper, we focus on the presence of passive RFID tags.

1.2 Characteristics of RFID Data

Because of the nature of RFID data, which is significantly different from traditional data warehouse approaches, characteristics of RFID data have to be fully considered in RFID data management systems.

Despite of diversity of RFID applications, RFID data share common fundamental characteristics [5], which can be listed as follows:

Simplicity of data: Data generated from an RFID application, can be seen as a stream of RFID observations of the form (*tag_id*, *reader_id, timestamp*), where *tag_id* and *reader_id* refer to EPCs which universally unique identify the tagged item and the RFID reader (as well as its location), and the timestamp is the time when the reading occurred respectively.

Large volume of data: One of the biggest concerns of RFID is to deal with a huge amount of information. The volume and velocity of RFID data will exceed the capacity of existing technology infrastructure. As an example, Wal-Mart is expected to generate 7 terabytes of RFID data per day [8]. Since each tag is tagged periodically and the data about tag EPC, location and reading time will be continuously produced in the system, even modest RFID deployments will generate gigabytes of rapidly changing data a day.

Temporal and dynamic: RFID observations are dynamically generated and the data carry state changes [20]. All RFID observations are associated with the timestamps, the objects' locations and the containment relationships change along the time. Thus, it is essential in RFID data management to model all such data in an expressive data suitable for application level interactions, including tracking and monitoring.

Implicit semantics and inaccuracy of data: Although the accuracy of current RFID observations is improving and METRO expects the raise of read rate accuracy up to 95-100 % [18], RFID data can be still considered as generally inaccurate. The observed read rate accuracy in real RFID deployments remains still on average in the

60-70 % range [13], which is one of the major factors limiting widespread adoption of RFID technology. Erroneous readings, such as missed or duplicate readings, have to be semantically processed. Moreover, primary RFID observations need to be automatically transformed by a framework into business logic data.

1.3 Principles of Effective RFID Data Management

In [16] Mark Palmer suggests seven common principles of the effective RFID data management which will ensure reliable, real time, accurate decisions. The first principle of Mark Palmer is called *"Digest the Data Close to the Source"*. This principle says that central IT systems should be protected from flooding of the huge amount of low-level data from RFID readers, it would be much better if somewhere before this information could be processed, filtered, and aggregated. The second principle is about transformation simple events into more complex meaningful events. Complex Event Processing, pioneered by David Luckham of Stanford University, deals with the task of processing multiple streams of simple events with the goal of deriving more complex meaningful events which infer from the simple events. Together with the third principle, called *"Buffering Event Streams"*, these three principles encompass the task field for RFID middleware and event processing. In the next section of our paper, we consider a system architecture for managing RFID data, the event processing issue will be discussed in Section 4. Two more principles of Mark Palmer, namely *"Cashing Context"* (required for determining complex events from simple RFID events data) and *"Federate Data Distribution in Near Real Time"*, are related to the principles mentioned above. The principle "*Graceful Aging RFID Data"* deals with one of the biggest problems for RFID applications - huge amount of generated RFID data. The last principle *"Automate Exception Handling"* can be considered as the primary job of any RFID system. It should be always kept in mind if we want to achieve the efficient RFID data management, but this principle is outside the scope of our research.

This paper is organized as follows. Section 2 discusses a system architecture for managing RFID data. Then, in Section 3 we consider efficient techniques for filtering RFID data. Section 4 addresses the issue of the event processing and details the usage of RFID middleware. Constructing of the RFID Warehouse we discuss in Section 5. Related work is discussed in Section 6, followed by concluding remarks.

2 System Architecture

Regarding the principles of effective data management, especially the first principle of Mark Palmer, the layered system architecture can be proposed for managing RFID data. Figure 1 illustrates this architecture. The lowest level consists of RFID tags attached to different items. The next layer, called *Data Capture layer*, is the layer of RFID readers. The data emerging from this layer can be considered as RFID data streams. As we already discussed in the previous section, RFID data stream can be described as a stream of tuples of the form (*tag_id*, *reader_id*, *timestamp*). In our model *reader_id* represents the RFID reader location. Typically, RFID readers emit
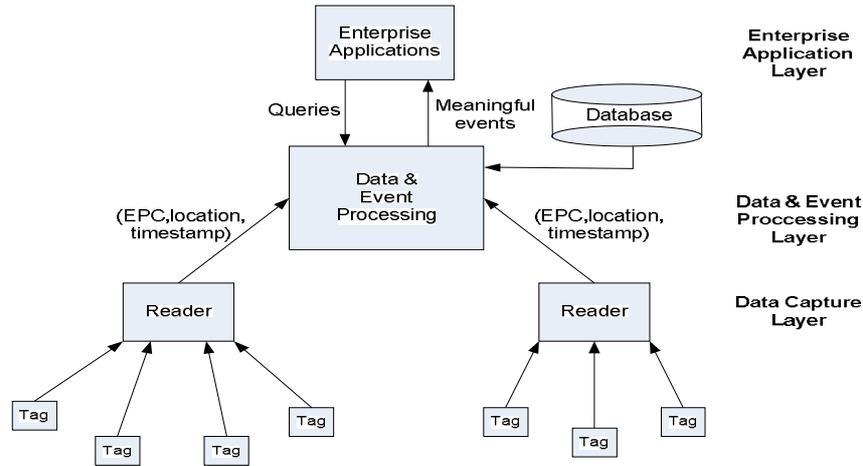
Fig. 1. System architecture

such tuples in response to some events, such as a timer expiring or a signalling from a motion-sensor when a new object has arrived or an object moves to a new location. A reader's mode of operation and the resulting frequency, with which it emits data, are often configurable [4]. This configuration of RFID readers can be ruled from the upper layer. Anti-collisions techniques and other low-level communication details between these two first levels are specified by the interface between them and the RFID protocols for this interface. Although connectivity technologies should be also taken into account when managing RFID data, the focus of this paper is the part of the system architecture that lies above the data capture layer.

The third layer of the architecture, *Data and Event Processing layer*, in [4] called *Savant,* is subject to a standardization effort under the name "middleware". This layer plays the primary role in RFID data management, and, thus, is the main point of our research. RFID middleware systems typically deployed between the readers and the applications in order to correct captured readings and provide clean and meaningful data to application logic. Thus, this layer is responsible for mapping the low-level data incoming from RFID readers to a more manageable form suitable for the interactions on the enterprise application level. Although RFID readers may be capable of simple filtering the stream of tuples, we are interested in more advanced data filtering, cleaning, and aggregation techniques performed on this layer.

Generally, there are three primary motivations for using RFID middleware: providing connectivity with RFID readers (via the reader adapter), processing raw RFID observations for consumption by applications (via event manager), and providing an application-level interface to manage readers and capture filtered RFID events [7]. In Section 4 we discuss RFID middleware more in details. Describing RFID data concentrator in [16], Mark Palmer suggests to include in the third layer the module *"In-memory Cache"* according to his third principle of effective RFID data management. But, for the sake of simplicity we do not consider this module in our architecture.

Applications on the enterprise application level can interact with the middleware layer by issuing simple queries as well as by installing standing queries that result in a stream of matching data. This level supports the business processes of enterprise applications such as Supply Chain Management (SCM) or Customer Relationship Management (CRM).

3 Filtering RFID Data Streams

This paper does not intend to describe all known RFID data cleaning techniques. In this section, we analyze the common problems with RFID data that often happen in RFID applications. First, we formalize the possible undesirable scenarios that lead to unreliable data. And then, we focus on the effective and efficient solutions for the problem of the redundancy elimination.

Missed and unreliable readings appear very often in RFID applications. The reasons are low-power, low-cost constraints of RFID tags, and not totally reliable wireless communication. Generally, these unreliable readings can be formalized in three typical undesired scenarios: *false negative readings, false positive readings* and *duplicate readings* [1], explained as follows:

- *False negative readings.* In this scenario, RFID tags, which are in the vicinity of a reader, might not be detected by this reader at all. This situation can be caused by i) RF collisions and signal interferences when multiple tags are to be simultaneously observed; ii) metal shielding, other physical obstructions or RF interference.

- *False positive readings (or noise).* In this case, in addition to RFID tags to be read, unexpected extra readings are generated. This can be caused by unknown reasons from the environment or the readers, for example, if one of the readers periodically sends wrong tag IDs.

- *Duplicate readings.* Duplicate observations are very common in RFID applications. They can be caused by several reasons: i) tags in multiple reading frames (in the scope of a reader for a long time) are read by the reader multiple times; ii) tags in the overlapped areas are read by multiple readers because these readers are installed to cover larger area or distance; iii) multiple tags with the same EPC are attached to the same object in order to raise reading accuracy and reliability.

If readings are performed in multiple cycles, their recognition rate can be increased [19]. In this case, noisy readings (false positive readings) generally have a low occurrence rate in comparison to normal true readings. Accordingly, the rate of false negative readings can be significantly reduced. However, this can produce much more duplicate readings [1]. In this paper, we focus on the redundancy problem. The solutions for this problem will be discussed at two different levels:

- Redundancy at data level
- Redundancy at RFID readers level

3.1 RFID Data Redundancy Filtering

Now we want to consider effective and efficient data filtering techniques to clean RFID data, which can be further integrated into RFID applications. First, we discuss the efficient technique for the false positive reading elimination, called *smoothing* or *denoising*, based on the *sliding window* approach proposed in [1]. Then, we show how this algorithm can be used for duplicates elimination (duplicates merging).

Considering readings in multiple cycles discussed above, only those readings are supposed to be true that repeat within a certain time interval with a rate higher than a predefined *threshold.* The sliding window is used to define this time interval. This window can be considered as a window with certain size (denoted as *window_size*) that moves in time. Therefore, if at an initial time moment *t* this window looked like $\left[t, t + window\_size\right]$, after λ, the coordinate of the window will become $\left[t + \lambda, t + window\_size + \lambda\right]$. Now, RFID readings in form of tuples (*tag_id, reader_id, timestamp*) will enter the sliding window. And to determine and eliminate noise readings, we need to count for each incoming reading how many readings with the same EPC (*tag_id*) value appear within the time window of size *window_size*. If this count is below a noise *threshold,* this reading is treated as noise and discarded. Otherwise all true observed readings are to be forwarded for further processing. Let us notice that in the simple case the search key of readings within the sliding window is usually (*tag_id, reader_id*). However, if tags are observed by multiple RFID readers, the search key can be *tag_id.*

Smoothing algorithm [1].

Two parameters are predefined: *window_size* for the sliding window size and *threshold* for the noise detection, both are configurable[1]. All incoming readings are maintained in the same FIFO (First-In, First-Out) queue (a window buffer). For each incoming reading *R* a full scan of the preceding sliding time window of size *window_size* is performed. If readings with same search key appear more than *threshold* times within the window, they recognized as non-noise. Thus, we output all such readings in the window including the incoming reading, and, to make sure, a particular reading is never output more than once, for each of them we set the *output_flag* Meanwhile, all expired readings are to be removed from the window buffer, expiration time can defined as (*INCOMING_READING.timestamp – window_size*).

Basically, the algorithm consists of four operations: inserting the incoming reading into the window, removing expired readings from the window, the reading counting with the same key, the setting of *output_flag* for the readings which satisfy the threshold condition. And after evaluating the time complexity for each operation, it is easy to see that the overall time complexity of the algorithm is $\Theta(n)$, where *n* is the number of readings in the sliding window [1].

---

[1] In most cases, the value for *window_size* parameter is defined based on the repeat count of a tag and the interval between repeats. The *threshold* parameter is usually tuned based on error rates.

The one drawback with the presented algorithm is that, it does not preserve the output order of the readings. The original order, where the first observed tagged object will be output first, may be critical for many RFID applications, for example according to medical procedures [1]. Thus, it is very important that this ordered is not changed after the filtering. Unfortunately, the smoothing approach allows a situation where this order can be broken. To solve this problem some improvements for the smoothing algorithm were proposed in [1].

Now we are coming to the problem of the duplicate readings. We assume that RFID stream has been already processed with the discussed smoothing algorithm for the noise elimination. Concerning duplicates, duplicate readings for the same tag have to be recognized, and only the earliest one among them should be retained and forwarded for further processing. Thus, to merge duplicate readings we need one more parameter – *duplicate_distance*. If a reading is within a *duplicate_distance* according to the previous reading with the same key, this reading is recognized as a duplicate and must be discarded. Otherwise, it is output as a normal new reading. In such a way duplicates readings can also be eliminated. It is obvious, we can easily realize the described merge algorithm by setting in the smoothing algorithm the size of the sliding window to *duplicate_distance*. In order to increase the efficiency of the merging algorithm, instead of performing a linear window scan, we can use a hashtable [1].

3.2 Redundant Reader Elimination in RFID Systems

The redundant RFID readers cause redundant wireless communication. This problem is related to the situation when a tag is tagged by more than one RFID reader simultaneously. The purpose here is to determine the minimal subset of the readers that can cover all tags. These readers may stay active, the rest is considered as redundant and can be safely simultaneously turned off without risk of the information loss. The temporary deactivation of the redundant readers will not reduce the number of tags covered by the initial reader network. Thus, in such a way the area monitoring accuracy will be increased by excluding redundant wireless communication. Besides, if there is no instant energy support, we can also longer the limited battery life of the RFID readers.
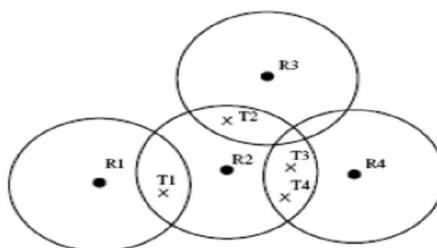

Fig. 2. Redundant RFID reader example

8

The problem of determining the coverage redundancy is not new for wireless sensor networks [23]. However, the solutions for these networks are not acceptable for RFID systems because of several aspects. The first reason, in RFID systems the coverage is defined in terms of discrete points (tags), not in terms of contiguous circular areas. Second, the assumption of the existence of location information or the ability to estimate distances between adjacent sensors creates the basis for the solutions in wireless sensor networks. Such an assumption is not reasonable in RFID systems.

Figure 2 illustrates the simple example for redundant RFID readers [3]. Four RFID tags T1, T2, T3, T4 on the figure are observed by four RFID readers R1, R2, R3, R4. The optimal solution requires only R2 to be active while three other readers can be turned off. A simple solution for this problem in RFID systems could be a simultaneous broadcasting a query containing empty string by all readers. Since all receiving tags must reply to such a query, a reader that receive no reply is treated as redundant. This solutions has two drawbacks. First, it requires time synchronization between all RFID readers. Second, there can be left uncovered tags that were previously covered by at least two redundant readers. In [3], Carbunar *et al* proposed for the redundant reader elimination problem a randomized, decentralized, and localized approximation algorithm, called RRE. Although RRE assumes the presence of passive tags only, it is applicable to any number of RFID readers and tags and makes no assumptions on the underlying reader or tag topology. One significant assumption for this approach is that, RFID tags have limited memory, part of which is writable.

RRE consists of two different steps. In the first step, each reader attempts to write the number of covered tags (its tag count) to all its covered tags. Therefore, first it must identify its tag count. It was proposed in [3] to use RCA algorithm for avoiding reader collisions and allowing RFID readers accurately detect tags in their vicinity. We may notice that discussed above smoothing algorithm can be also used for this purpose. After each reader identified its tag count, it sends a write command containing its reader identifier and tag count to all its covered tags. An RFID tag may store only the highest value seen for tag count, along with the identity of the corresponding reader.

In the second step, an RFID reader queries each of its covered tags. Each tag replies with stored information, and, if the reader receives from a tag its own ID that means it locked this tag and is responsible for the monitoring of this tag. A reader that has locked no tag can be considered as redundant and safely turned off.

Some problems arise with this approach. The first is synchronization problem. Since in the example R_4 covers only two tags, it can complete RCA and respectively the fist step of RRE before R_2 that covers four tags, and, despite its redundancy stay active. To solve this problem all active readers can maintain a list of locked tags and listen passively for RFID tag replies to queries. And if an RFID reader receives a message that its locked tag has another holder with a higher tag count than its own, it removes this tag from its list of locked tags. Second, the algorithm assumes that the position of readers does not change for a long period of time. This assumption may not be held in applications such as supply chain where the reader position may change in order to optimize the business process [4]. A solution for this problem can be a periodical reactivation of inactive readers and execution RRE on all readers. Finally to

this approach, even with centralized knowledge of the RFID system topology, an optimal solution for the redundant reader elimination problem is NP-hard [3].

4 Event Processing and RFID Middleware

One of the most challenging and interesting issues in RFID data management is to transform the low-level data into more sophisticated manageable form which can be useful for the enterprise applications. The goal is to transform RFID data management system into a real-time and interactive system. In order to achieve this, different levels of inferences must be done on raw data [4].

Regarding the second principle of the effective RFID data management, complex event detection is one of the primary roles played by the *Data* and *Event Processing* layer of the system architecture. Generally, RFID readings can be considered as simple events. In order to derive from them more complex events which are semantically appropriate to end-user applications, a declarative event-based framework is required. This framework will allow to determine the relationship among simple and complex events by describing the specifications of a particular application business logic with the declarative statements or rules. In this sense, the work of RFID data processing can be greatly simplified, and the cost of RFID data integration can be significantly reduced [21]. The declarative event-based framework is discussed next.

4.1 Complex Event Processing

There are generally two types of RFID applications [21]:
 − *History oriented object tracking.* RFID data streams emerging from multiple RFID readers at distributed locations are transformed into semantic data stored in RFID data store. The RFID data store essentially preserves the history of the movement and behaviors of objects. And as shown in [4], RFID observations and their collected data are highly temporal. Thus, beside the location (geographic or symbolic such as a warehouse) and the relationship among objects (containment relationship in simple case), the semantics of the data include temporal nature of RFID data.
 − *Real-time monitoring.* In real-time applications implying special application logic RFID events can trigger real-time responses. An example for this application type can be a security system for a firm which should send an alert if it detects an item tag (i.e. tag on a laptop) leaving the building without detection (within a defined time interval, i.e. 10 sec) a tag of the authorized employee.

To process RFID data and make them suitable for the enterprise applications, we need two steps: first, the application logic must be interpreted as complex events, and, then, such complex events must be detected. Traditional event systems do not support temporal characteristics of RFID events, and therefore, they cannot be applied to RFID applications. Both the temporal distance between two events and the interval of a single event are critical for event detection. Thus, a framework is required to proc-

ess complex RFID events, which is different from traditional event systems and capable of taking into account all peculiarities of RFID events. To build this scalable rule-based system, we need to provide the semantics formalization for RFID events and rules. In our events and rules formalization we based on [21].

4.1.1 RFID Event Semantic Formalization

An event can be either a primitive event or a complex event. Primitive events occur at a point in time, while complex events are patterns of primitive events and happen over a period of time. In the following formalization E represents event type, and e represents event instance. For the following discussion we need several functions in our event expression which are defined as follows:

- *t_begin(e),* returns the starting time of the event instance e
- *t_end(e),* returns the ending time of the event instance e
- *interval(e) = t_end(e) - t_begin(e),* returns the interval of an event instance e
- *dist(e1,e2) = t_end(e2) - t_end(e1),* returns the distance between two event instances e_1 and e_2
- *interval(e1,e2) = max{ t_end(e2), t_end(e1) } - min{ t_begin(e2), t_begin(e1)}* , returns the interval between two event instances e_1 and e_2

Primitive events are atomic, a primitive event either happens completely or does not happen at all. As we discussed in the previous sections, the primitive RFID events are RFID observations or tuples of the form (*tag_id*, *reader_id, timestamp*). According to the tag EPC (*tag_id*) or reader EPC (*reader_id*), the primitive events can be of different types. Two user-defined functions can be used to define primitive event types:

- *group(reader_id)* – the group which the reader belongs to
- *type(tag_id)* – type of the object with EPC *tag_id*

The primitive event type E can be specified now as*:*
$E = observation$(*tag_id*, *reader_id, t*), *group(reader_id)=*'gr' , *type(tag_id)=*'PC'
That means, observations of 'PC' by readers in group "gr" are of type E.

In order to construct complex events, RFID *event constructors* are used. They are of two types: non-temporal and temporal, the latter contains order, temporal constrains or both. A complex event can be build now by applying event constructors to its constituent events, which are either primitive events or other complex events.

Only three basic non-temporal complex event constructors can be defined to express all event patterns without temporal constrains: *OR, AND, NOT.* For example, a complex event E = ALL(E_1, E_2, E_3), which means that all three events must occur, is not more than E = E_1 AND E_2 AND E_3.

Temporal event constructors for the specification of temporal complex events is the essential part of the rule-based event systems associated with RFID applications, which differs these systems from traditional event approaches. The following set of temporal event constructors can be proposed [21]:

- *SEQ(;)*: Sequence of two events, the second event occurs under the condition that the first event has already occurred

- *TSEQ(*E1*;* E2, t1, t2)*:* Distance-constrained sequence of two events where the temporal distance between the occurrences of E1 and E2 is bounded by [t1, t2]:
  t1≤ dist(E1, E2) ≤ t2

- *SEQ$^+$ (E):* The aperiodic sequence operator, allows to express one ore more occurences of an event *E*

- *TSEQ$^+$ (* E, t1, t2 *):* The distance-constrained aperiodic sequence operator

- *WITHIN(E, t):* An interval-constrained event, occurs if an instance of *E (e)* occurs and *interval(e)* ≤ t

As an example, we can specify the complex event for the security system mentioned above:

WITHIN(E1 AND NOT(E2), 10 sec), where

E1 = *observation*(*EPC1*, *reader_id, t1),* type*(EPC1)*='laptop'

E2 = *observation*(*EPC2*, *reader_id, t2),* type*(EPC2)*=' authorized employee'.

4.1.2 RFID Rules

Based on described above event specification, RFID rules can be defined using the following syntax :

```
CREATE RULE rule_id, rule_name
        ON event
       IF condition
     DO action1; …;actionn
```

where *condition* is a boolean combination of user-defined functions and SQL queries; action1; …; actionn is an ordered list of actions.

In [21] is shown how declarative RFID rules can provide powerful support for data processing, including data filtering, data transformation and aggregation, and real-time monitoring. Associating relationship among objects has been identified as a difficult issue for RFID applications [20]. This problem can be solved with this rule-based approach. For example, we want to define the rule for the containment relationship aggregation. A feasible description of the semantic for this complex event would be: if a distance-constrained aperiodic sequence of readings from reader "*reader_1*" is observed followed by a distinct reading from a reader "*reader_2*", it implies that all objects observed by "*reader_1*" are being packet in the object observed by "*reader_2*". The rule can be specified as follows:

```
DEFINE E1 = observation(EPC1, reader_1, t1)
DEFINE E2 = observation(EPC2, reader_2, t2)
CREATE RULE r, containment_rule
ON TSEQ(TSEQ⁺(E1, 0.1 sec, 1 sec); E2, 10sec, 20sec)
IF true
DO BULK INSERT INTO CONTAINMENT VALUES(EPC2,EPC1,t2,"UC")
```

where "*UC*" stands for "until changed", CONTAINMENT is a table for the containment relationship information, and *BULK* will enforce insertion of all contained objects into the container.

4.1.3 RFID Event Detection

After the application logic was interpreted as complex events, we should consider an effective algorithm for RFID event detection. RFID Complex Event Detection Algorithm (RCEDA) was proposed in [21]. This algorithm uses a graph-based computation model. Figure 3 demonstrates an example of graphical representation of complex events used in RCEDA [21]. A graph representing the events for a set of rules is constructed in three steps:

1. An event graph for each rule's event is built, where leaf nodes represent primitive events.
2. Interval constrains are propagated.
3. Common sub-graphs are merged.
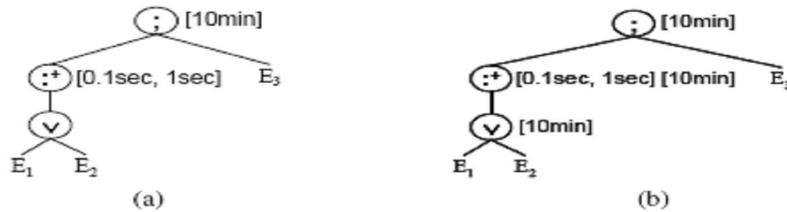
Fig. 3. Graphical representation of an interval-constrained complex event E=WITHIN($TSEQ^{+}$ (E1 AND E2, 0.1sec, 1 sec); E3, 10 min): (a) before propagating the interval constraint; (b) after propagating the interval constraint.

Wang *et al* showed in [21] that traditional graph-based event processing systems, which detect events in a bottom-up fashion (occurrences of primitive events are injected at the leaves and flow upwards to trigger parent complex events), are not applicable to detecting RFID events. The reason is non-spontaneous events which often occur in RFID systems (such as generated from NOT constructors) – events that cannot detect their occurrences by themselves unless they get expired, in case, if they are associated with interval constrains, or are explicitly queried from their parent nodes. To support the detection of non-spontaneous events, *pseudo events* are proposed in [21]. They are automatically generated by the system to actively trigger the querying of the occurrences of these non-spontaneous events.

Thus, the set of data transformation and aggregation rules allows RFID low-level data to be automatically interpreted and mapped into their data models and stored in RFID data store. Additionally to low-level data filtering discussed in Section 3, this declarative framework supports semantic data filtering which extracts data on demand or interprets semantics from RFID data. The simple example for semantic data filtering, associated with *infield* and *outfield* events used in smart shelf applications [18], was given in [21]. With our declarative framework we can describe all possible combinations among simple and complex RFID events, taking into account temporal nature of RFID events. The task of RFID applications is only to generate proper patterns

for complex events. For a fully automatic environment, where human intervention may not be possible, we can configure the order of readings among objects, by properly deploying readers and the workflow. One of the main advantages of RFID rule-based framework is that, data transformation and aggregation is greatly simplified in a declarative way.

4.2 RFID Middleware

Effective RFID middleware is a growing concern to end users of RFID in the supply chain because middleware can ensure the accuracy of inventory data and validity of RFID-tagged items. There are many commercial implementations of RFID middleware available today [2], [14], [16], [17] . In our paper, we focus on Siemens RFID Middleware [20]. The rule-based solution described in the previous section is fully integrated into Siemens RFID middleware. The formalized for data transformation rule-based framework is depicted in Figure 4 [20].

Siemens RFID middleware relies on *Dynamic Relationship ER Model (DRER)*. DRER includes the following static entities: *SENSOR, OBJECT, LOCATION*, and *TRANSACTION*. State-based dynamic relationships include: *SENSORLOCATION* generated from *SENSOR* and *LOCATION*, *OBJECTLOCATION* generated from *OBJECT* and *LOCATION, CONTAINMENT* generated from *OBJECT* and itself; Event-based dynamic relationships include: *OBSERVATION* generated from *OBJECT* and *SENSOR*, and *TRANSACTIONITEM* generated from *OBJECT* and *TRANSACTION* [20]. The structure of the *OBJECTLOCATION* table is illustrated in Table 1. This table preserves the location history, two attributes *tstart* and *tend* are associated with a state-based dynamic relationship, to represent the lifespan of that relationship, "UC" denotes "until now".
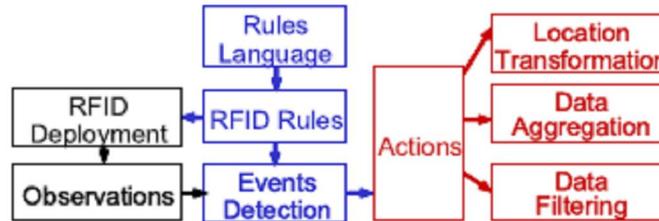

Fig. 4. Rule-based RFID data transformation

Table 1. Sample OBJECTLOCATION

EPC	Location_id	tstart	tend
1.1.1	L001	2006-04-25 15:22:00.000	2006-04-25 20:12:00.000
1.1.2	L002	2006-05-10 10:00:00.000	2006-05-15 12:17:00.000
1.1.2	L002	2006-05-18 11:05:00.000	UC

The simplicity of DRER is that, it requires minimum extensions to ER model, by simply adding two new types of relationships, each of which is associated with special time-related attributes. But, despite its simplicity, this temporal model enables the powerful support for complex RFID queries. These queries generally are divided into two categories: *RFID Object Tracking* and *RFID Object Monitoring*. We can see on an example for object monitoring, how easy we can obtain the required information: if we want to find items sold to customers in the last hour, the query can be specified:

```
SELECT epc FROM OBJECTLOCATION
WHERE location_id = 'POS_ID' AND tend = 'UC' AND
tstart ≤ sysdate-(1/24)

where POS_ID is RFID reader at POS (point-of-sale).
```

Bornhoevd *et al* developed in [2] Auto-ID Infrastructure which converts RFID data into the data for business processes, running on either SAP or non-SAP backend systems, with specified mapping rules and metadata. However, we had no intention to provide a comparison analysis between Siemens and SAP RFID middleware specifications.

One of the most challenging problems in RFID applications is the problem how to manage a large volume of accumulated data in a reasonable way. The active life period of RFID objects, during which these objects are tracked and monitored, is normally limited. The question arises: which data should be actively retained for further processing and which data can be safely compressed and archived. To solve this problem, the efficient partitioning method integrated into Siemens middleware is proposed in [20]. This method considers two scenarios: fixed period partition and dynamic period partition. The fixed period partition, which assumes that objects have close active life periods, can not be effectively applied to RFID applications, where the active life periods are irregular. In this case, the dynamic period partition is more suitable. This method proposes to partition data dynamically according to *active object ratio (AOR)* as the number of active objects over the total number of objects in the active partition (an object becomes inactive if it finishes its movement cycle) [20]. Then a minimum AOR_{min} is defined below which archiving will be performed. With the updates of the data in the active partition P_{active}, the AOR will decrease, until it reaches the threshold AOR_{min}. Then, all objects in P_{active} will be archived into partition P_i associated with a partition interval. Then updates will be continuously performed on active objects in P_{active} and the process repeats. Thus, with partitioning, updates are only performed on the active partition, and most queries are performed on the active partition as well, thus they are more efficient. For historical queries the search space can be narrowed down to fewer partitions according to the partition intervals.

5 RFID Warehousing

In the presence of terabyte-sized date new data structures and algorithms are required that can provide fast responses to different queries. In this section we discuss several issues assigned with the architecture and the constructing of the RFID warehouse.

The first problem that should be considered before we develop an RFID warehouse is a large design problem [4]. In practice, determining number, type, and placement of readers, and the manner in which they are connected to other sensors is part of a large design problem. This will affect the implied size of higher-level infrastructure to support generated data rate from the readers. One should decide often between state-based design and event-based design. State-based design, where items are periodically tagged and monitored, produces much more data. However, by event-based design, where the system reacts only on meaningful events such as enter a new good onto a shelf, some options are not available because the current location of an item may be out of readers' range. There is no universal solution for this problem. An optimal solution will rely on the application business logic, and, while the hardware configuration is difficult to change on frequent basis, it must be economically feasible.

Although the task of gathering a stream of raw data from tag readers to the centralized database has many commonalities with analogous tasks in data warehousing [4], there are some significant differences. In contrast to RFID, in a typical data warehousing setup (e.g., for a department store), it is uncommon for the data to be queried at its source (e.g., the point-of-sale terminal). Besides, in Section 4 we showed the necessity of applying different levels of inferences on raw data to transform RFID data management system into a real-time and interactive system. While in traditional data warehouses we select a set of views to materialize in order to facilitate the query processing time, in RFID data management systems, we must decide which set of inference rules should be materialized beforehand (eager approach) and which set should materialize at the query time (lazy approach or *Inference Rule Materialization*) [5]. Thus, due to the nature of RFID, *uncertainty of inference rules* and *regular update on data* make *inference rule materialization* in RFID warehouse significantly different from *materialized-view maintenance* in traditional data warehouses.

Indeed, if an item that had to be packed into pallet is not detected in warehouse on this palette arrival, we must consider various possible explanations. These different interpretations are the results of uncertain inference rules. Moreover, in contrast to traditional data warehouses where updates are desired only infrequently at designed times, currency of data is very important in a typical RFID scenario. Besides, the problem with the form of data aggregation in traditional warehouse is that the links between records are not considered. If we want, for example, to know how many items traveled from a particular distribution center, we will have difficulties to get this information.

To solve these problems and to manage successfully RFID data a new data structure was proposed in [8], [10], called *RFID-Cuboid*, for storing aggregated data in the RFID warehouse. The underlying compression principles based on two observations [8]:

– items usually move together in large groups through early stages in the system (e.g., distribution centers) and only in later stages (e.g., stores) they move in smaller groups;

– although RFID data is registered at the primitive level, data analysis usually takes place at a higher abstraction level. And since many users are only interested in data at a relatively high abstraction level, data compression can be explored to group, merge, and compress data records.

The *RFID-Cuboid* consists of three tables:
1. *Info*, which stores product information for each RFID tag
2. *Stay*, which stores information on items that stay together at a location
3. *Map*, which stores path information necessary to link multiple stay records.

The main distinctive feature of this approach is the presence of the *map* table linking records from the fact table (*stay*) in order to preserve the original structure of the data.

As shown in [8], the *RFID-Cuboids* form the basis for future query processing and analysis. The advantage of these data structures is that they aggregate and collapse many records in the cleansed RFID database. The efficient algorithm for constructing an *RFID-Cuboid* was proposed in [8]. However, the details of this algorithm, as well as the problems of the pre-computation of *RFID-Cuboids* at different levels and construction of higher level *RFID-Cuboids* are outside the scope of our research.

6 Related Work

The recently emerging RFID technology poses new challenges in data processing and management systems. Harrison summarizes in [11] the data characteristics of RFID data and provides some reference relationships to represent the data, but his model is not effective on supporting complex queries such as RFID object tracking and monitoring.

The layered system architecture for managing RFID data was proposed in [5]. But Derakhshan et al consider data capture layer responsible for providing different filtering and cleaning procedures, while we suppose this level to be limited in its abilities and focus on the next level, data and event processing layer, as more powerful for providing low-level data filtering as well as semantic data filtering.

RFID data filtering is essential to provide accurate data used in RFID applications. In [13] Jeffery *et al* introduced SMURF (*Statistical Smoothing for Unreliable RFID Data*), the first declarative, adaptive smoothing filter for cleaning raw RFID data streams. SMURF models the unreliability of RFID readings by viewing RFID streams as a statistical sample of tags in the physical world, and exploits techniques grounded in sampling theory to drive its cleaning processes.

Event processing has been studied in [9], [22]. Wu et al proposed *SASE* as the complex event processing system and the expressive language to support queries for complex event processing.

Sophisticated RFID platforms, including Siemens Middleware [20], Sybase RFID Solutions [14], [17], SAP Auto-ID Infrastructure [2], are provided recently by major IT vendors. These systems represent a general interface to collect low-level RFID data from readers and then forward them to applications. However, they mostly support limited RFID rules, only primitive events or their simple combinations.

Masciari proposed in [15] a framework for dealing with the outlier (or anomalies) detection problem, which is very challenging in RFID warehousing. However, this problem remains an open issue and can be a scope for future research.

7 Conclusion

RFID applications are set to play an essential role in object tracking and supply chain management systems. In this paper, we identify the problems of RFID data management, filtering and aggregation. Our specifications based on common principles of effective RFID data management. Although these principles did not provided any methodology, they pointed out the importance of the event processing and were the basis for future research.

Starting with the characteristics of RFID data, we formalized the logical way of transforming low-level RFID observations into semantically meaningful data suitable for different enterprise applications. In our research, we followed this way from the layered system architecture for managing RFID data, presented in Section 2, over the discussed in Sections 3 and 4 solutions for filtering and data aggregation problems to the usage of RFID middleware.

In our paper, we focused on the Data and Event Processing layer of the architecture, standardized as middleware. This layer is typically deployed between the readers and the applications in order to apply to captured data efficient cleaning, filtering, and aggregation techniques, and, then, forward clean and meaningful data to application logic. On example of Siemens RFID Middleware we showed how the efficient techniques for data filtering and aggregation discussed in the previous sections can be integrated into RFID middleware.

The problem of RFID data filtering was considered in Section 3 in two different aspects: data redundancy and RFID reader redundancy. After the formalization of possible undesirable scenarios which lead to unreliable data, we discussed the efficient algorithms that provide solutions for each of these two problems.

In Section 4 we described the rule-based declarative framework for the event processing that can greatly simplify RFID data transformation and aggregation in a declarative way. We showed the advantages of this framework, and, meanwhile, we discussed why traditional event-based approaches are inapplicable in RFID environment.

The last point of our research was the discussion of specific problems associated with inferencing and RFID Warehousing in Section 5. We described differences between RFID Warehousing and traditional approaches. Then we presented *RFID-Cuboids* as the basis for future query processing and analysis.

However, there is still a number of aspects that might be subject of further research. The problem of graceful aging of RFID data is one of the biggest problems in RFID applications. The proposed in Section 4 partitioning method cannot be considered as the universal solution for this problem. An interesting open research problem would be using lineage tracing to find the appropriate set of inference rules to materialize under the consideration of uncertainty of inference rules. Also the problem of the outlier detection remains still open.

References

1. Bai, Y., Wang, F., Liu, P..: Efficiently Filtering RFID Data Streams. In CleanDB Workshop (2006) 50-57
2. Bornhoevd, C., Lin, T., Haller, S., Schaper, J.: Integrating Automatic Data Acquisition with Business Processes – Experiences with SAP's Auto-ID Infrastructure. In VLDB (2004) 1182-1188
3. Carbunar, B., Ramanathan, M., Koyuturk, M., Hoffmann, C., Grama, A.: Redundant-Reader Elimination in RFID Systems. Sensor and Ad Hoc Communications and Networks (2005) 176-184
4. Chawathe, S., Krishnamurthy, S.,Ramachandrany, S., Sarma, S.: Managing RFID Data. Proceeding of the international conference on VLDB (2004) 1189-1195
5. Derakhshan, R., Orlowska, M., Li, X..: RFID Data Management: Challenges and Opportunities. IEEE International Conference on RFID 2007, Grapevine, Texas, USA (2007)
6. Finkenzeller, K.: RFID Handbook: Fundamentals and Applications in Contactless Smart Crads and Identification. John Wiley & Sons (2003)
7. Glover,B., Bhatt, H.: RFID Essentials. O'Reilly Media, Inc. First Edition (2006)
8. Gonzalez, H., Han, J., Li, X., Klabjan, D.: Warehousing and Analyzing Massive RFID Data Sets. 22nd IEEE ICDE Conference (2006)
9. Gyllstrom, D., Wu, E., Chae, H., Diao, Y., Stahlberg, P., Anderson, G.,: SASE: Complex Event Processing over Streams. In Proceedings of the Third Biennial Conference on Innovative Data Systems Research, CA, January (2007)
10. Han, J., Gonzalez, H., Li, X., Klabjan, D.: Warehousing and Mining Massive RFID Data Sets. 2006 Int. Conf. on Advance Data Mining and Its Applications (ADMA'06), China, August (2006)
11. Harrison, M.: EPC Information Service – Data Model and Queries. Technical Report, Auto-ID Center, October (2003)
12. Heinrich, C.: RFID and Beyond. Growing Your Business Through Real World Awareness. Wiley Publishing, Inc, Indianapolis, Indiana, USA (2005)
13. Jeffery, S., Garofalakis, M.,Franklin, M.: Adaptive Cleaning for RFID Large Data Bases. Proceeding of the 32nd international conference on VLDB (2006) 163-174
14. Manage Data Successfully with RFID Anywhere Edge Processing. iAnywhere Solutions, Inc, Dublin, USA (2005)
15. Masciari, E.: RFID Data Management for Effective Objects Tracking. SAC 2007 (2007) 457-461
16. Palmer, M.: Principles of Effective RFID Data Management. Enterprise Systems (2004)
17. Sybase RFID Solutions, http://www.sybase.com/rfid (2005)
18. The METRO Group Future Store Initiative. http://www.future-store.org, http://www.eweek.com/article2/0,1895,2152562,00.asp
19. Vogt, H.: Efficient Object Identification with Passive RFID Tags. In Pervasive (2002)
20. Wang, F., Liu, P.: Temporal Management of RFID Data. Proceeding of the international conference on Very large data bases (VLDB) (2005) 1128-1139
21. Wang, F., Liu, S., Liu, P., Bai, Y..:Bridging Physical and Virtual Worlds: Complex Event Processing for RFID Data Streams. EDBT 2006 (2006) 588-607
22. Wu, E., Diao, Y., Rizvi, S.: High-Performance Complex Processing over Streams. SIGMOD Conference 2006 (2006) 407-418
23. Zhang, H., Hou, J.: Maintaining Coverage and Connectivity in Large Sensor Networks. In International Workshop on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless and Peer-to-Peer Networks, Februar (2004)