Temporal Management of RFID Data

Fusheng Wang Peiya Liu

Integrated Data Systems Department
Siemens Corporate Research
755 College Road East, Princeton, NJ 08540, USA
{fusheng.wang, peiya.liu}@siemens.com

Abstract

RFID technology can be used to significantly improve the efficiency of business processes by providing the capability of automatic identification and data capture. This technology poses many new challenges on current data management systems. RFID data are time-dependent, dynamically changing, in large volumes, and carry implicit semantics. RFID data management systems need to effectively support such large scale temporal data created by RFID applications. These systems need to have an explicit temporal data model for RFID data to support tracking and monitoring queries. In addition, they need to have an automatic method to transform the primitive observations from RFID readers into derived data used in RFID-enabled applications. In this paper, we present an integrated RFID data management system – Siemens RFID Middleware – based on an expressive temporal data model for RFID data. Our system enables semantic RFID data filtering and automatic data transformation based on declarative rules, provides powerful query support of RFID object tracking and monitoring, and can be adapted to different RFID-enabled applications.

1 Introduction

1.1 Background

RFID (radio frequency identification) technology uses radio-frequency waves to transfer data between readers and movable tagged objects, thus it is possible to create a physically linked world in which every object is numbered, identified, cataloged, and tracked. RFID is automatic and fast, and does not require line of sight or contact between readers (or sensors) and tagged objects. With the significant advantages of RFID technology, RFID is being gradually adopted and deployed in a wide area of applications, including supply chain management [1, 2, 3, 4, 5], retail [6], anti-counterfeiting [7], security [8], and healthcare. For example, Siemens Business Services launched a pilot project to track patients with RFID bracelets during hospital admissions [9].

Through the automatic data collection, RFID technology can achieve greater visibility and product velocity across supply chains, more efficient inventory management, easier product tracking and monitoring, reduced product counterfeiting and theft, and much reduced labor cost. On the other hand, there is a chasm between the physical world and the interpreted world through sensor observations. These observations need to be automatically interpreted and semantically transformed into business logic data, before they can be integrated into business applications, such as ERP and WMS.

1.2 Characteristics of RFID Data and Problem Statements

Despite the diversity of RFID applications, RFID data share common fundamental characteristics, which have to be fully considered in RFID data management systems.

Temporal and dynamic. RFID applications dynamically generate observations and the data carry state changes. All sensor observations are associated with the timestamps when the readings are made; objects' locations change along the time; the containment relationships change along the time, and all EPC [10] related transactions are also associated with time. (EPC – Electronic Product Code – is an identification scheme for universally identifying physical objects, defined by standard committees [10].) It is essential to model all such information in an expressive data model suitable for application level interactions, including tracking, monitoring, and application integration [11, 12].

Implicit semantics and inaccuracy of data. In an RFID system, a reader observation comprises of the reader EPC, the observed EPC value of an RFID tag, and the timestamp when the observation occurs. These data carry implicit information, such as changes of states and business processes

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, requires a fee and/or special permission from the Endowment.

**Proceedings of the 31st VLDB Conference,
Trondheim, Norway, 2005**

(e.g., change of locations), and further derived information such as aggregations (e.g., containment relationship among objects). Thus, a framework is needed to automatically transform observations into business logic data.

Meanwhile, while the accuracy of current RFID readers is improving, in current RFID applications, there are still erroneous readings, such as duplicate readings or missing readings. Such erroneous data have to be semantically filtered.

Streaming and large volume. RFID data are generated quickly and automatically, and accumulated for tracking and monitoring. The data volume generated can be enormous, which requires a scalable storage scheme, to assure efficient queries and updates.

Integration. RFID data have to be integrated with existing applications for product tracking and monitoring. This requires an RFID data management system be easily configured to be integrated into different applications, with minimum integration cost.

In early RFID solutions, sensor readings are directly sent to applications and services, thus it is up to the applications to interpret the preliminary readings, and generate business logic data. This approach has much complexity on RFID data interpreting, and is not scalable and adaptable.

Recently, major IT vendors are moving quickly to provide RFID solutions [13, 14, 15, 16, 17]. These systems provide sophisticated middleware-based platforms for RFID applications, and serve as bridges between the physical RFID world and the rest software infrastructure. High level data modeling, on the other hand, is up to applications.

1.3 Our Approach to RFID Data Management

In this paper, we propose an expressive temporal-based data modeling of RFID data, using a *Dynamic Relationship ER Model (DRER)*. By maintaining the history of events and state changes, the data model captures the fundamental RFID application logic into the data model itself, thus complex queries can be easily supported. With a rules-based framework, observations can be automatically transformed into business logic data, through user configured rules. By bringing all the technologies together, we build *Siemens RFID Middleware*, an integrated RFID data management system. Salient features of our approach include:

- *Expressive Data Model.* DRER is built on ER model, with minimum extensions. The semantics of the data model is generalized from RFID data, thus fits perfectly with RFID data.
- *Effective Query Support.* Based on the temporal data model, common RFID queries such as RFID object tracking and monitoring, and other complex queries, can be effectively supported. Moreover, partitioning-based archiving assures efficient updates and queries.
- *Automatic Data Acquisition and Transformation.* With a framework of rules-based transformation, the system achieves automatic data filtering and transformation, with minimum configuration.
- *Adaptable and Portable.* The business-logic integrated data model and the rules-based data transformation can be adapted to different RFID applications, and substantially reduce the cost of managing RFID data and integrating RFID data into business applications.

This paper is organized as follows. In Section 2, we propose the Dynamic Relationship ER Data Model for RFID data, then we show in Section 3, how the data model can benefit queries in an RFID data management system. In Section 4, we discuss how to semantically process RFID data and automatically transform RFID data into business logic. Then, in Section 5, we show how to efficiently support queries and updates through partitioning-based archiving. Next in Section 6, we discuss Siemens RFID Middleware that integrates all the technologies into an adaptable system. Section 7 discusses experiments and Section 8 discusses related work. Section 9 concludes the paper.

2 RFID Data Modeling

2.1 Motivation

A Sample RFID-enabled Application. Figure 1(a) shows a simplified example of an RFID-enabled supply chain system. In a supplier warehouse, each product item is EPC-tagged. (Here we assume products are tagged at case level. Our methods can be generalized to any tagging level.) The tagged cases are packed onto a pallet at the supplier warehouse (location L001), where the EPC tags of both the cases and the containing pallet is automatically scanned by an RFID reader R1. (Here we assume all RFID tags are passive tags, which are the cheapest and most commonly used in current RFID applications.) Then, at the warehouse loading zone, pallets are loaded into a truck, and both the pallets and the truck are scanned by another reader R2. The truck then departs to a retail store, through a predefined route (location L002). At the unloading zone of the retail store (location L003), all pallets are unloaded from the truck, and all cases are unpacked from the pallets. All objects, the truck, pallets and cases, are scanned by another reader R3, and then the cases are stocked in the store. Eventually, when cases are purchased by customers (location L004), they are scanned by reader R4 at the register. In the whole process, observations from RFID readers are made automatically, and data are also collected automatically.

One key concept in the above RFID application is *location* [18]. A location can be a geographic location or a symbolic location. Here we assume symbolic location, which can be a warehouse, a shipping route, a surgery room, or a smart box [19]. Location changes come with the movement of objects and business processes. Another key concept is *aggregation*, such as *containment*. Containment determines a hierarchical relationship among objects. For example, a pallet is loaded with cases, or a toolbox contains a set of tools [20]. The containment relationship implies important logic information, for example, a truck leaving a warehouse implies that all its contained objects leave the warehouse too.

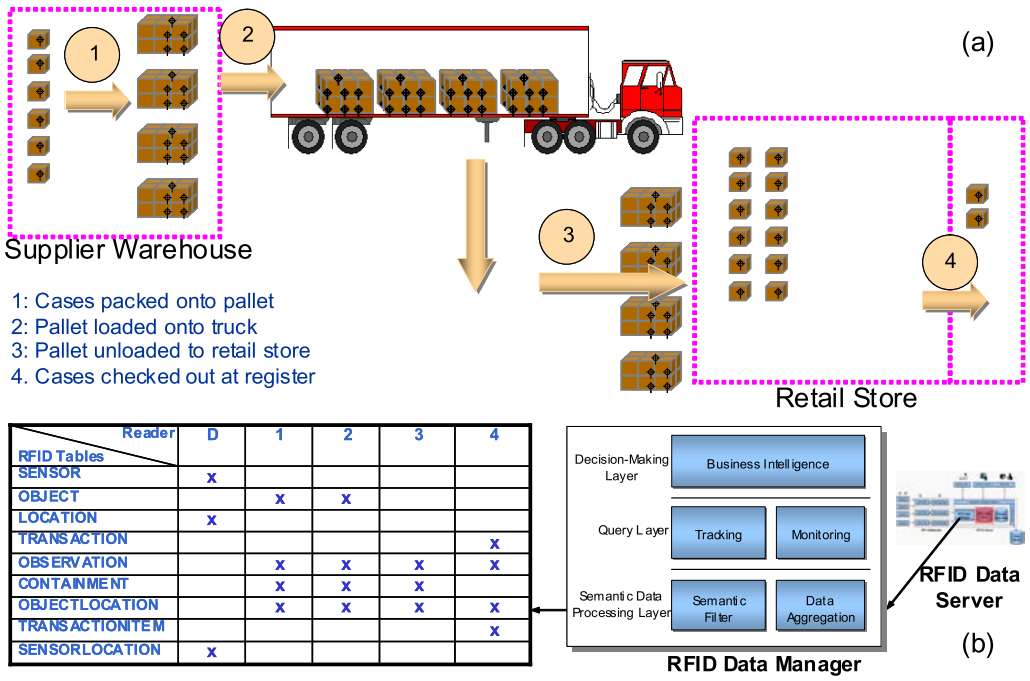

Figure 1: A Sample RFID-Enabled Supply Chain System

We also observe that RFID data are highly dynamic, and the data always carry state changes. On the other hand, objects are associated with a unique ID encoded in EPC, and these IDs are static and non-alterable.

Based on these observations, we propose a temporal-oriented data model for RFID data, by first identifying the fundamental entities, as discussed next.

2.2 Fundamental Entities in RFID Systems

In an RFID system, there are two basic categories of data: *static data* and *dynamic data* [21, 22]. Static data are related to commercial entities and product/service groups, such as location information, product level and serial level information. Dynamic data are specific to individual items. There are two types of dynamic data: *instance data* such as serial number and date of manufacture, and *temporal data* such as observations, location and containment changes of objects, which are all captured through EPC-tag readings. Among all the data, the temporal data are directly related to the fundamental business logic in RFID applications, such as the movement and transaction of products, and are crucial for an RFID data system to track and monitor objects.

By examining RFID systems, we summarize the following primary entities that interact with each other and generate business processes.

– *Objects.* These include all EPC-tagged objects, such as items, cases, pallets, trucks, even patients with RFID-bracelets. For example, the objects in Figure 1(a) include cases, pallets, and trucks.

– *Sensors/Readers.* RFID readers use radio-frequency signals to communicate with EPC tags and read the EPC values. We have readers R1, R2, R3 and R4 (marked with circles) in Figure 1(a). Each RFID reader (or its antenna) is also uniquely identified by its EPC code. In this paper, we use "reader" and "sensor" interchangeably.

– *Locations.* A location is symbolized to represent where an object is/was. It can be a warehouse, a retail store, a distribution center, or a route between two locations. For example, in Figure 1(a), the locations include: warehouse(L001), route from the warehouse to a retail store(L002), retail store (L003) and customers (L004). In a smart box application [19], locations are simply "in-box" and "out-of-box". The granularity of locations can be defined according to application needs. A location is also associated with an owner.

– *Transactions.* There can be business transactions in which EPC is involved. For example, a checkout involves a credit card transaction with many EPC readings. Transactions are business-specific, i.e., with different transaction types. Transactions are not considered in many RFID applications.

Next we discuss how these entities interact with each other.

2.3 Dynamic Interactions between RFID Entities

While the above discussed entities are themselves static in the business processes, they dynamically interact with each other and generate movement, workflow, and business logic. These interactions generate events and states changes. State changes include:

– *Object location change.* For instance, the truck and its loaded pallets leave the warehouse.

– *Object containment relationship change.* For example, cases are packed onto pallets, as shown in Figure 1(a).

– *Reader location change.* For instance, a reader is deployed at a new location.

There are also another state change, ownership change, for example, cases unloaded into the retail store. This can be combined with location changes, since a location is always associated with an owner. The information about during which period an object is in certain state is essential and has to be captured.

Besides state changes, there are also events generated in the interactions, including:

– *Observations.* These are generated when readers interact with objects.

– *Transacted items.* These are generated when an object participates into a transaction.

All events are associated with timestamps when they occur.

In a summary, the entities have dynamic relationships with each other, which lead to events and states changes. However, current RFID data management systems only store and manage events, and state information is implicit and has to be derived. This makes it very complicated to search state information. For example, based on the event-based model proposed in [22], it takes 8 steps for the query: "where are the 5 CDs that were supposed to be in the last order?".

One of the essential goals of an RFID-enabled application is to track objects and monitor the system at any location, at any time, or both. This requires an expressive data model that can explicitly represent the history of both events and states, and capture fundamental business logic into the data model itself, therefore complex queries such as tracking and monitoring can be effectively supported.

In the following, we propose a Dynamic Relationship ER Model (DRER) that fits perfectly with RFID data.

2.4 Dynamic Relationship ER Model (DRER)

DRER is a temporal extension of ER model. In ER model, all entities and relationships are static or current. In an RFID system, entities are static, but all the relationships are dynamic. Here we naturally extend ER model by inheriting all the ER model semantics and simply adding a new relationship – dynamic relationship, as shown in Figure 2. (For simplicity, other attributes are ignored in the figure.) There are two types of dynamic relationships: relationship that generates events, and relationship that generates state history. Here we use dash lines to represent state-based dynamic relationship, and dash-dot lines to represent event-based dynamic relationship.

Two attributes **`tstart`** and **`tend`** are associated with a state-based dynamic relationship, to represent the lifespan of that relationship; and an attribute **`timestamp`** is associated with an event-based dynamic relationship, to represent the occurring timestamp of the event.

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.

Thus, based on DRER data model, we have following static entities for RFID data (Figure 2): **`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`**.

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

We use symbol 'UC' to denote *now*. 'UC' can be represented as 'end-of-time', e.g., "9999-12-31 23:59:999", in the database. In the remainder of this paper, our granularity for time is a millisecond, thus two continuous events have a difference of 1 millisecond. All the techniques we present are equally valid for any granularity used by an application.

The tables for DRER model are described as follows.

(Static) Entity Tables

`SENSOR(sensor_epc, name, description)`

The SENSOR table records the EPC, name and description of a sensor (Table 1). [1]

sensor_epc	name	description
1.255.1	R1	Packing reader
1.255.2	R2	Departure reader
1.255.3	R3	Unloading reader
1.255.4	R4	Checkout reader

Table 1: Sample SENSOR

`OBJECT(epc, name, description)`

The OBJECT table includes the EPC, name, and description of an EPC-tagged object (Table 2).

epc	name	description
1.1.101	Case	Containing items
1.1.102	Case	Containing items
1.1.103	Case	Containing items
1.1.104	Case	Containing items
1.2.1	Pallet	Containing cases
1.3.1	Truck	Loaded with pallets

Table 2: Sample OBJECT

`LOCATION(location_id, name, owner)`

[1]Here we use a pseudo notation of EPC code for illustration purpose. The format of EPC is defined in [10].

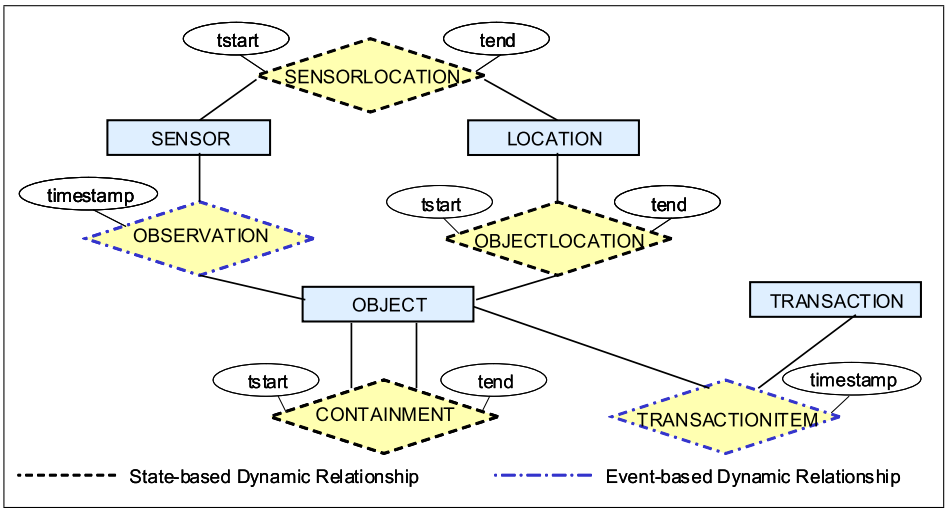

Figure 2: Dynamic Relationship ER Model

location_id	name	owner
L001	Warehouse A	Supplier A
L002	Route to retail C	Carrier B
L003	Retailer store C	Retailer C
L004	Customer	Customer

Table 3: Sample LOCATION

transaction_id	transaction_type
TX00001	retail_checkout
TX00002	retail_checkout
TX00003	retail_checkout
TX00004	retail_checkout

Table 4: Sample TRANSACTION

The location table defines symbolic business locations used for tracking, including id, name, and owner of a location (Table 3).

`TRANSACTION(transaction_id, transaction_type)`

Since transaction data are business specific, here we simplify a transaction as a record with transaction id and a transaction type (Table 4).

Here we only consider the fundamental attributes. If additional attributes are needed, it can be extended with an extension table. For example, for the SENSOR table, there can be an extension table: **`SENSOR_EXT(epc, property, value)`**.

Dynamic Relationship Tables

`OBSERVATION(sensor_epc, value, timestamp)`

This table records the raw reading data generated from sensors (i.e., readers), including sensor's EPC, tag's EPC value, and the reading timestamp (Table 5).

`CONTAINMENT(epc, parent_epc, tstart, tend)`

This table records in what period **`[tstart, tend]`** an object (identified by its EPC) is contained in a parent object (identified by its parent EPC) (Table 6).

sensor_epc	value	timestamp
1.255.1	1.1.1	2004-11-01 10:33:00.100
1.255.1	1.2.1	2004-11-01 10:34:00.000
1.255.2	1.2.1	2004-11-01 10:35:00.001
1.255.3	1.1.1	2004-11-07 11:00:00.001
1.255.4	1.1.1	2004-11-08 15:30:00.010

Table 5: Sample OBSERVATION

epc	parent_epc	tstart	tend
1.1.1	1.2.1	2004-11-01 10:33:00.100	2004-11-07 11:00:00.000
1.1.2	1.2.1	2004-11-01 10:33:00.110	2004-11-07 11:00:00.010
1.2.1	1.3.1	2004-11-01 10:35:00.001	2004-11-07 10:59:00.000

Table 6: Sample CONTAINMENT

`OBJECTLOCATION(epc, location_id, tstart, tend)`

This table preserves the location history of each object, including an object's EPC, location id, and the period **`[tstart, tend]`** during which the object stays in that location (Table 7). (*UC* denotes 'now' in the examples.)

epc	location_id	tstart	tend
1.1.1	L001	2004-10-30 17:33:00.000	2004-11-01 10:35:00.000
1.1.1	L002	2004-11-01 10:35:00.001	2004-11-07 11:00:00.000
1.1.1	L003	2004-11-07 11:00:00.001	2004-11-08 15:30:00.009
1.1.1	L004	2004-11-08 15:30:00.010	UC

Table 7: Sample OBJECTLOCATION

`SENSORLOCATION(sensor_epc, location_id, position, tstart, tend)`

This table keeps the location history of a sensor, since a sensor can be deployed at different locations/positions. It includes the EPC of a sensor, location id, position, and the period during which the sensor is located in that location. Here **`position`** is a symbolic position of a sensor at a location, for example, the loading zone of a warehouse. A location of a sensor at certain time can always be found in this relation.

`TRANSACTIONITEM(transaction_id, epc,`

sensor_ epc	location_id	position	tstart	tend
1.255.1	L001	packing	2004-10-01 15:30:00.010	UC
1.255.2	L001	loading	2004-10-01 15:30:00.010	UC
1.255.3	L003	unloading	2004-10-01 15:30:00.010	UC
1.255.4	L003	register	2004-10-01 15:30:00.010	UC

Table 8: Sample SENSORLOCATION

transaction_id	epc	timestamp
TX00001	1.1.1	2004-11-08 15:30:00.010
TX00001	1.1.2	2004-11-08 15:30:00.100
TX00001	1.1.3	2004-11-08 15:30:00.109

Table 9: Sample TRANSACTIONITEM

timestamp)

This table keeps the items in a transaction. It includes a transaction id, EPC of the object in the transaction, and the timestamp when the object's transaction occurs.

3 Tracking and Monitoring RFID Data

A significant benefit of the temporal modeling is the power to support complex RFID queries. Most RFID queries are temporal queries with temporal constraints such as history, temporal snapshot, or temporal slicing. There are more complex ones such as temporal joins and temporal aggregates. We summarize these queries as two main categories: *RFID Object Tracking* and *RFID Object Monitoring*. The former is to track RFID objects including missing objects. RFID Object Monitoring is to monitor the states of RFID objects and the RFID system. In the following, we summarize common RFID data tracking and monitoring types, and propose methods to express such queries based on DRER data model.

3.1 Methods for RFID Data Tracking

RFID Object Tracking tracks the change history of an object's states and detects missing objects.

RFID Object Tracking

Q1. Find the location history of an object with EPC value 'EPC'.

```
SELECT * FROM  OBJECTLOCATION
WHERE epc='EPC'
```

Missing RFID Object Detection

There are two scenarios for Missing RFID Object Detection, the first one is *Missing RFID Object Tracking*, to locate when and where an object was lost, knowing the lost object's EPC. This means that the object appeared at previous locations, but not at current location.

Q2. Find when and where object 'MEPC' was lost.

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

The second scenario is *Possible Missing RFID Object Searching*, to search if there is any missing object at a certain location C, knowing that at a previous location L and timestamp T, all objects were complete. This can be done by comparing the two sets of objects between location C and location L.

Q3. Check if there are missing objects at current location C, knowing that all objects were complete at previous location L at time T.

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

RFID Object Identification

Since every RFID object is uniquely identified by its EPC, it is easy to identify an object:

Q4. A customer returns a product with EPC 'XEPC'. Check if this product was sold from this store (location 'L003').

```
SELECT *
FROM  OBJECTLOCATION
WHERE epc='XEPC' AND location_id='L003'
```

RFID Object Moving Time Inquiry

One common query is to find how long it takes for an object to move from one location to another.

Q5. How long did it take to supply object 'OEPC' from location S to location E?

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

3.2 Methods for RFID Data Monitoring

RFID Object Monitoring is to monitor the states of RFID objects and the RFID system. These include snapshot inquiry, temporal slicing inquiry, temporal join query, temporal aggregation, and containment examination.

RFID Object Snapshot Query

By specifying a snapshot timestamp, it is easy to monitor the snapshot information of any RFID objects, including snapshot locations, containment, observations, or transactions.

Q6. Find the direct container of object 'EPC' at time T.

```
SELECT parent_epc
FROM   CONTAINMENT
WHERE  epc='EPC' AND
       tstart <= 'T' AND tend >= 'T'
```

RFID Object Temporal Slicing Query

This query retrieves object information during a temporal interval.

Q7. Find items sold to customers in the last hour.

```
SELECT epc FROM OBJECTLOCATION
WHERE location_id = 'L04' AND tend = 'UC'
 AND tstart <= sysdate-(1/24)
```

RFID Object Temporal Join Query

Temporal join query will retrieve information by joining multiple relations on certain temporal constraint.

Q8. This case (with epc 'TEPC') of meat is tainted. What other cases have ever been put in the same pallet with it?

```
SELECT c2.epc
FROM CONTAINMENT c1, CONTAINMENT c2
WHERE c1.parent_epc = c2.parent_epc
 AND c1.epc = 'TEPC' AND overlaps(
 c1.tstart, c1.tend, c2.tstart, c2.tend)
```

where **`overlaps()`** is a user-defined scalar function to check if two intervals overlap. User-defined scalar temporal functions can be defined to simplify temporal queries.

Temporal Aggregation of RFID Data

This query will summarize aggregational information at certain snapshot or interval.

Q9. Find how many items loaded into the store 'L003' on 11/09/2004.

```
SELECT count(epc)
FROM OBJECTLOCATION
WHERE location_id = 'L003'
 AND tstart <= '2004-11-09 00:00:00.000'
 AND tend  >= '2004-11-09 00:00:00.000'
```

RFID Object Containment Queries

RFID containment queries are queries that retrieve the containment relationships between RFID objects. These queries are normally interleaved with other temporal RFID queries. Two special cases are recursive containment queries: *RFID Object Sibling Search*: find all the sibling objects of a container object, and *RFID Object Ancestor Search*: find all the ancestor container objects of an object. The following shows an example of sibling search (ancestor search can be done similarly by switching parent and child attributes).

Q10. RFID Sibling Object Search. Find all objects contained in object 'PEPC'.

```
WITH RECURSIVE all_sub(parentepc, epc) AS
 (SELECT parentepc, epc
  FROM CONTAINMENT
  WHERE parentepc = 'PEPC'
  UNION
  SELECT a.parentepc, c.epc
  FROM all_sub a, CONTAINMENT c
  WHERE a.epc = c.parentepc
 )
 SELECT *
```

Based on the DRER schema, we can also specify constraints and business intelligence queries, such as automatic shipping notice, low inventory alert, and trend analysis.

The RFID temporal queries can also be expressed with standard temporal query languages, such as TSQL2 [23], which, however, is not yet supported by commercial RDBMS.

4 Automatic RFID Data Acquisition and Transformation

4.1 Data Acquisition from Physical World

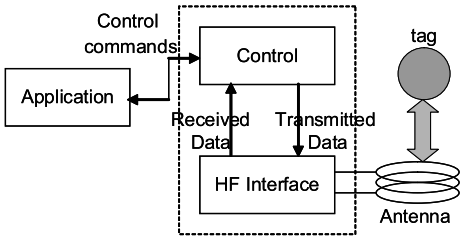

Figure 3: Data Flow of RFID Readings [24]

An RFID-enabled system provides automatic data acquisition through RFID readings. The data flow of RFID readings is shown in Figure 3 [24]. A reader consists of two blocks: control block and high frequency (HF) interface. The application sends control commands to the control block of the reader, and then the control block controls the HF interface to read RFID tags. The received data are sent back to the application through the control block. There can be two modes for an RFID reader: *full duplex/half duplex mode* (or *inventory mode*), and *sequential mode*. In the first mode, when the reader RF field is turned on, the response of a tag is broadcast. This mode is recommended for applications that expect multiple tags detection at once. For the sequential mode, the reader RF is switched off briefly at regular intervals, and the tag recognizes the gap and responds to the reader.

4.2 Rules-based RFID Data Transformation

In order to effectively track and monitor RFID objects, the acquired data need to be automatically transformed into high level semantic data, through: i) *Data Filtering*. The observations from readers may contain errors such as duplicates and have to be filtered; ii) *Location Transformation*. RFID observations can imply change of locations and business movement, and need to be interpreted and represented. iii) *Data Aggregation*. There can be semantic relationships among RFID objects, such as containment relationships. Such relationships are implicit and have to aggregated according to the observation patterns. Transformed data are represented in DRER model, and stored in RFID data store (more will be discussed in Section 6).

While such data transformation can be performed at the application integration level, it poses much complexity on data integration, and is not adaptable and difficult to scale.

Next, we propose a rules-based framework for automatic RFID data transformation.

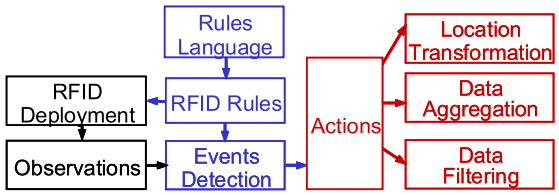

Figure 4: Rules-based RFID Data Transformation

RFID data streams consist of primitive events and composite events, which can serve as the base for data transformation. Here we formalize a rules-based framework to automate the transformation, as shown in Figure 4. First, RFID readers and their operations are defined to follow certain patterns at the deployment, according to application models. According to this, users can define active rules on observation streams to detect events. Primitive events, such as observations from readers, can lead to actions such as change of locations. Sequence events consist of a sequence of primitive events of the same type, defined by the order and closeness of intervals. Composite events are a combination of the above, and can lead to actions such as aggregation of data. Next we briefly discuss the rules-based framework for automatic data transformation.

Data Filtering

This will filter data according to predefined constraints with global and domain information. For example, multiple readers can generate duplicate readings. To filter this, a filter will scan data within a sliding window to find if there are duplicate EPC readings from multiple readers, and delete the duplicate if it exists. As an example, the following rule specifies that if readings from reader **Rx** and **Ry** have the same EPC value within **T**, then one of them is dropped.

```
OBSERVATION(Rx, e, Tx),
OBSERVATION(Ry, e, Ty), Rx <> Ry,
within(Tx, Ty, T)
  -> DROP:OBSERVATION(Rx, e, Tx)
```

Location Transformation

One fundamental transformation is to transform RFID sensor observations into location changes. For instance, Reader "R2" is mounted at warehouse departure zone and will scan objects before their departure. A rule for this transformation is defined as follows:

```
OBSERVATION("R2", e, t) ->
   UPDATE:OBJECTLOCATION(e,"L002", t, "UC")
```

The above rule defines that any observation generated from reader "R2" will change the observed object's location, i.e., delete object **e**'s current location by updating the ending timestamp of current location to **t**, and insert a new location "L002" for this object, with starting timestamp **t**, and ending timestamp *UC* (*Until Changed*).

Data Aggregation

Events can be aggregated into semantic data, such as containment relationships. Associating relationship among objects has been identified as a difficult issue for RFID applications [25]. This problem can be solved with our rules-based approach. For a human intervened environment, aggregation can be done by manually generating a start event and an end event for the aggregation, which then can be detected by the rules system. 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. Then we use rules to detect such ordering to generate the aggregation relationship. For instance, when pallets are loaded into a truck to depart, a sequence of readings on the pallets are done, followed by (with a distinctive distance) a separate reading of the truck's EPC. This sequence of events will aggregate as a containment relationship between the pallets and the truck, and lead to updates in the **CONTAINMENT** table. This can be automated by a rule based on composite events defined as follows:

```
seq(s,"r2");OBSERVATION("r2", e, t) ->
  INSERT:CONTAINMENT(seq(s, "r2", Tseq),
    e,t,"UC")
```

seq(s,"r2") defines a sequence event from reader "r2" with maximum adjacent distance **Tseq**. When a sequence is followed by another observation of the same reader, all the objects will be put as children of **e** in the **CONTAINMENT** table.

Based on the predefined rules, an event detector (Figure 4) monitors the observation streams to detect events, and automatically triggers actions to generate logic data, as discussed in Section 4.3.

Thus, to deploy the RFID data management system to different applications, we only need to declare the rules and constraints, which significantly reduces code rewriting and make the system configurable and adaptable.

4.3 Data Generation from Rule Actions

The data transformation rules described above will lead to actions to generate logic data represented in DRER in the database. We can categorize them into following scenarios according to the triggering event types:

- Data generation triggered from reading events. These include updates on tables **OBSERVATION**, **OBJECTLOCATION**, **CONTAINMENT**, and **OBJECT** (when an object first appears).
- Data generation triggered from business transaction events. These include updates on the **TRANSACTION** table.
- Data generation triggered from both reading events and business transaction events. These include updates on **TRANSACTIONITEM** table.

Besides, there are also data generated at RFID deployment. These include updates on **LOCATION** table, **SENSOR** table and **SENSORLOCATION** table. Such updates are mostly manually updated and are infrequent.

We can further classify the actions as two types: insertion of new events, and modification of states:

- Insertion of a new event. A new record is inserted in the corresponding event table with the current timestamp.
- Insertion of a new state. This will create a new state record (e.g., a new record in **`OBJECTLOCATION`** table), with the starting timestamp **`tstart`** as the current timestamp, and ending timestamp **`tend`** as *UC*.
- Deletion of a state. A deletion of a state means the ending of a state. This can be done by changing the ending timestamp **`tend`** to the value of the current timestamp.
- Update of a state. This can be seen as a deletion followed by an insertion.

In particular, when a parent container object is updated with a location change, all its containing objects' locations will be updated recursively. (In practice, it may not be possible to scan tags contained inside a container.)

Figure 1(b) demonstrates the automatic data collection, data transformation, and data modification process. As objects move from the system through the four readers (1 to 4), corresponding tables (marked with 'x') are updated to reflect changes of states and occurring of new events. Reader 'D' is a pseudo reader to denote modifications occurred at deployment.

5 Efficient Query Support with Partitioning

With fast reading speed, RFID systems generate large volume of data. Accumulation of RFID data can lead to slower queries and updates. Nevertheless, RFID objects normally have limited active life period, during which the objects are actively updated, tracked and monitored. For example, in a supply chain system, an object starts from the time when it is first tagged and scanned, and ends when it is sold to customers. Therefore, we can partition and archive non-active RFID data, and perform most queries and update operations on active RFID data. This can achieve better query performance and also assure efficient updates. We propose two approaches of archiving for different scenarios: fixed period partition and dynamic period partition.

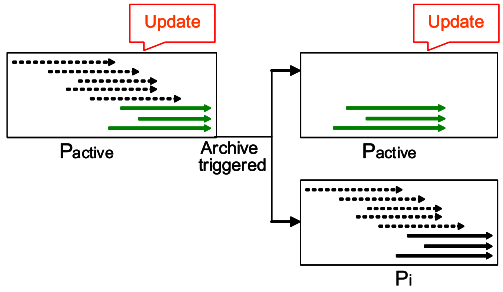

Figure 5: Data Archiving with Partitioning

5.1 Fixed Period Partition

Suppose all objects' behavior is homogeneous and they have close active life periods, e.g., T. We assume all active objects are stored in an active partition P_{active}. For every T period, we check the active partition, and move all objects from the active partition to an archive partition P_i associated with a partition interval: [P_{istart},P_{iend}]. Only active objects will be kept in P_{active} partition for future updates, as shown in Figure 5 (where solid lines represent active objects). The process is repeated every T time, and eventually it leads to a set of history partitions in addition to the active partition (P_1, P_2, P_3, ..., P_{active}).

5.2 Dynamic Period Partition

The fixed period partition works well when the active life periods are uniform and with limited length. This may not be true for some RFID applications, where the active life periods are irregular, or in the case finer partition is needed. In these cases, we propose to partition data dynamically according to active object ratio.

We first define *active object ratio (AOR)* as the number of active objects over the total number of objects in the active partition. An object is inactive if it finishes its movement cycle, or it is not current any more. We then define a minimum AOR_{min} 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; only active objects will be kept in P_{active}. Then updates will be continuously performed on active objects in P_{active}, until the AOR reaches AOR_{min}. The process is then repeated.

Both fixed and dynamic partitions can be physical partitions, or logical partitions associated with partition numbers, which can become part of the search keys. 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, e.g., time slicing and snapshot queries, the search space can be narrowed down to fewer partitions according to the partition intervals. Meanwhile, if required, data can also be quickly vacuumed by removing expired partitions.

6 Siemens RFID Middleware

In the previous sections, we discuss the RFID data model and the benefit of complex query support, rules-based automatic data transformation, and partitioning-based storage. All these technologies have to be put together to make them work. Next we discuss Siemens RFID Middleware, an RFID data management system that integrates all the technologies into a robust system. The objectives of the Siemens RFID Middleware is to provide an integrated solution for RFID applications, with automatic data acquisition, filtering and transformation; expressive data modeling and effective query support; and is adaptable to different applications with minimum configurations. Next we discuss the architecture of the system.

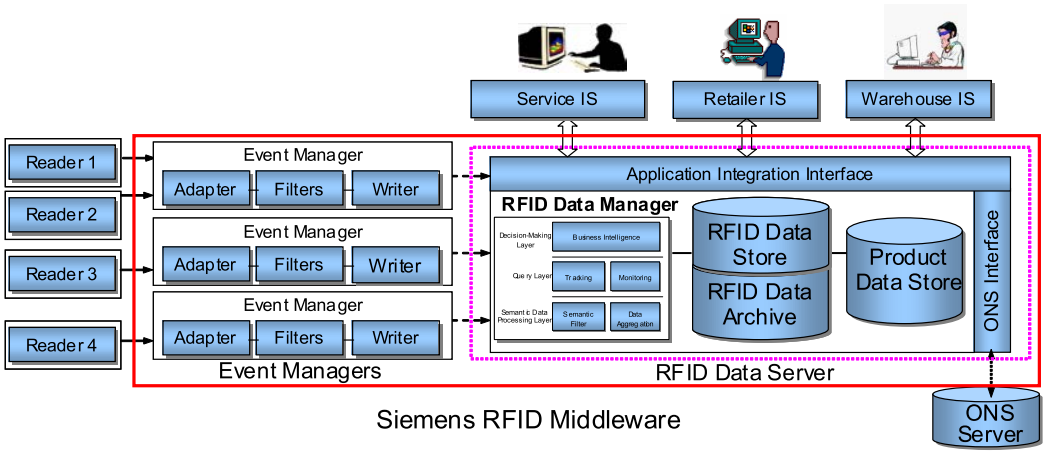

Figure 6: Siemens RFID Middleware

6.1 Siemens RFID Middleware Architecture

The Siemens RFID Middleware consists of the following components: RFID Readers, Event Managers, and RFID Data Server. The RFID Data Server includes RFID Data Manager, RFID Data Store, RFID Data Archive, Product Data Store, and integration interfaces. In the following, we discuss each component in the system.

Event Managers

Event Managers are the front end of the system, which dynamically receive data from readers, preliminarily filter the data, and forward the data to the RFID Data Server. Event Managers function in parallel at different clients. Each Event Manager can connect to multiple readers, and process the data generated from the readers simultaneously. An Event Manager includes Reader Adapters, Filters, and Writers.

Reader Adapter. Reader Adapter is the software component to communicate with RFID readers, and provides a unified interface for RFID middleware to access readings. A reader adapter can send commands to a reader to control a reader, such as reading frequency and cycle delay time, and also receive reading data.

Filter. Filter is the data filtering component to preliminarily screen raw reading data. The filtering functions can be duplicate removing from a reader, error detection, and so on. Since a filter in an Event Manager may not have global information, more advanced data filtering is performed at the Semantic Data Processing Layer in the RFID Data Manager discussed next.

Writer. Writer will format the data with PML (Physical Markup Language) [26], the standard language for RFID data exchange, and send them to different targets as messages, streams or other formats, through Web Services, JMS, HTTP response, or TCP/IP data packets.

RFID Data Manager

RFID Data Manager is the key component of RFID Data Server in an RFID Data Management System. It provides expressive data modeling, semantic data filtering, data aggregation, RFID object tracking and monitoring, and decision-making support. It consists of three layers: Semantic Data Processing Layer, Query Layer, and Decision-Making Layer, discussed as follows.

Semantic Data Processing Layer. This layer provides high level semantic data processing including semantic filtering and automatic data transformation and aggregation. A rules-based framework is formalized to automate the transformation.

Query Layer. This layer defines methods for RFID object tracking and RFID object monitoring, as discussed in Section 3.

Decision-making Layer. This layer provides business intelligence such as automatic shipping notice, low inventory alert, trend analysis, and so on.

RFID Data Store

RFID Data Store provides schemas implemented from DRER data model, and stores RFID data for RFID object tracking and monitoring, and decision-making. It also provides interfaces for data retrieval.

RFID Data Archive

By partitioning, non-active data are archived into history partitions in RFID Data Archive.

Product Data Store

Besides dynamic data stored in RFID Data Store, there are static information related to EPC objects, such as product description, product model, etc. Such information is preserved in the Product Data Store. (Stored data can be more general, such as patients, blood, etc.)

Data Integration

RFID Data Server provides an application integration layer to integrate the system with other applications. Product-level information is exchanged through Object Naming Service Server(ONS) [27]. Besides product level information, information at the object level (objects and their movement histories) can also be shared among different enterprises, for example, in an XML-based standard format, so the global history of an object can be combined and shared to monitor the global processes.

7 Experiments and Future Work

The prototype of Siemens RFID Middleware is developed and demonstrated at Siemens Corporate Research.

We have investigated the feasibility and effectiveness of Siemens RFID Middleware for RFID data management in health care, to increase healthcare safety and workflow efficiency. For example, for RFID-enabled healthcare asset management, major healthcare equipments, such as beds, wheelchairs, and medical devices, are RFID-tagged, so healthcare workers can locate any asset in real-time. This is extremely important for emergency room. RFID-enabled healthcare can not only achieve significant time saving, but also increases medical safety. We demonstrate that our system can effectively model healthcare asset movement, and provide real-time tracking of every asset item.

Currently, we are in the process of working with a hospital on implementing an RFID-enabled healthcare asset management based on the Siemens RFID Middleware.

We are also working with Siemens RFID Research Lab and Siemens MOBY[28] to provide streamlined RFID enabled solutions for warehouse/logistics, product assembly lines, transportation, and healthcare.

8 Related Work

RFID Data Modeling and RFID Platforms

RFID technology has emerged for years and poses new challenges for data management [11, 29, 30, 31]. However, little research has been done on how to effectively data modeling RFID data. Harrison et al [22, 32] summarize the data characteristics of RFID data, and provide some reference relations to represent the data. In their model, RFID data are modeled as events, thus the state history and the temporal semantics of business processes are implicit. This data model is not effective on supporting complex queries such as RFID object tracking and monitoring. A query often needs to be divided into numerous steps [22], which is indirect and inefficient, and not nature for users.

EPCglobal [33], the current EPC standard group, defines the networks for RFID data and product data [12, 21], but RFID data modeling is not a task in its working group.

Recently, major IT vendors are providing sophisticated RFID platforms, including the Sun EPC Network [14], SAP Auto-ID Infrastructure [13], Oracle Sensor Edge Server [15], IBM WebSphere RFID Premises Server [16], Sybase RFID Solutions [17], and Microsoft RFID Middleware [34]. UCLA's WinRFID Middleware [35] is another RFID middleware based on web services. These platforms serve as the bridges between the RFID physical world and the rest of the software infrastructure. RFID data are acquired, filtered and normalized through the platforms, and then dispatched to applications. Thus the high level RFID data modeling is up to applications.

There are also much work on using RFID in ubiquitous computing [19, 18]. Römer et al [18] emphasize the importance of location management, but no systematic model is proposed. Query processing in sensor data networks has been extensively studied in the past [36, 37], with a focus on solving problems under the power constraint of sensor networks.

Temporal Data Modeling

There has been much interesting work on ER-based [38] temporal modeling of information systems at the conceptual level. For instance, ER models have been supported in commercial products for database schema designs, and more than 10 temporal enhanced ER models have been proposed in the research community [39]. As discussed in the survey by Gregersen and Jensen [39], there are two major approaches of extensions to ER model for temporal support, devising new notational shorthand, or altering the semantics of the current ER model constructs. These approaches assume general temporal information, i.e., both entities and relationships, together with their attributes, are dynamic. They need significant extension of current ER data model, and the implementation can be also complicated. For the special case of RFID data, it has its own unique characteristics, and the modeling can be much simplified. The temporal data model proposed in this paper naturally fits with RFID data, and requires minimum extensions to ER data model.

9 Conclusion

RFID technology is significantly changing the current business applications. One of the major challenges for RFID technology is RFID data management [11, 29, 30]. In this paper, we have identified several key issues in managing RFID data, including data modeling, automatic data transformation and enrichment, effective support of queries for tracking and monitoring, scalable data archiving, and convenient application integration.

We propose a general and expressive temporal-oriented data model – Dynamic Relationship ER Data Model – for RFID data. This data model exploits the common characteristics of RFID data, and integrates business processes into the data model itself. The data model is shown to be quite powerful on supporting RFID data tracking and monitoring, which is essential for RFID applications. Our partitioning-based storage assures scalable data archiving and efficient updates and queries. The rules-based framework enables automatic RFID data filtering, transformation and aggregation, to generate high level semantic data. The Siemens RFID Middleware brings all these technologies together into an integrated RFID data management system. The system is general and can be adapted into different RFID applications, thus substantially reduces the cost of managing and integrating RFID data into business applications.

References

[1] Building the Digital Supply Chain: An Intel Perspective. http://www.intel.com/business/bss /industry/retail/digsuppchain.pdf, Janurary 2005.

[2] Walmart Supplier Information: Radio Frequency Identification Usage. http://www.walmartstores.com, 2005.

[3] World's Third Largest Retailer Completes Warehouse RFID Implementation. http://www.informationweek.com/story/showArticle.jhtml?articleID=57702741, January 2005.

[4] Tesco Pushes on with Full-scale RFID Rollout. http://www.computing.co.uk/news/1160636, Janurary 2005.

[5] DOD RFID Official Website. http://www.dodrfid.org.

[6] The METRO Group Future Store Initiative. http://www.future-store.org.

[7] Combating Conterfeit Drugs: A Report of the Food and Drug Administration. http://www.fda.gov/oc/initiatives/counterfeit/report02_04.html, February 2004.

[8] Homeland Security to Test RFID. http://www.rfidjournal.com/article/articleview/1360/1/1/, January 2005.

[9] Siemens to Pilot RFID Bracelets for Health Care. http://www.infoworld.com/article/04/07/23/HNrfidimplants_1.html, July 2004.

[10] EPC Tag Data Standards Version 1.1. Technical report, EPCGlobal Inc, April 2004.

[11] S. S. Chawathe, V. Krishnamurthy, S. Ramachandrany, and S. Sarma. Managing RFID Data. In *VLDB*, pages 1189–1195, 2004.

[12] S. Clark, K. Traub, D. Anarkat, and T. Osinski. Auto-ID Savant Specification 1.0. Technical report, Auto-ID Center, September 2003.

[13] C. Bornhoevd, T. Lin, S. Haller, and J. Schaper. Integrating Automatic Data Acquisition with Business Processes - Experiences with SAP's Auto-ID Infrastructure. In *VLDB*, pages 1182–1188, 2004.

[14] A. Gupta and M. Srivastava. Developing Auto-ID Solutions using Sun Java System RFID Software. http://java.sun.com/developer /technicalArticles/Ecommerce/rfid/sjsrfid/RFID.html, Oct 2004.

[15] Oracle Sensor Edge Server. http://www.oracle.com/technology/products/iaswe/edge_server.

[16] WebSphere RFID Premises Server. http://www-306.ibm.com/software/pervasive/ws_rfid_premises_server/, December 2004.

[17] Sybase RFID Solutions. http://www.sybase.com/rfid, 2005.

[18] K. Römer, T. Schoch, F. Mattern, and T. Dübendorfer. Smart Identification Frameworks for Ubiquitous Computing Applications. *Wireless Networks*, 10(6):689–700, December 2004.

[19] M. Lampe and C. Flrkemeier. The Smart Box Application Model. In *2rd International Conference on Pervasive Computing*, 2004.

[20] M. Lampe and M. Strassner. The Potential of RFID for Moveable Asset Management. In *Workshop on Ubiquitous Commerce at Ubicomp 2003*, 2003.

[21] The EPCglobal Network and The Global Data Synchronization Network (GDSN): Understanding the Information & the Information Networks. http://www.epcglobalinc.org/news/position_papers.html, October 2004.

[22] M. Harrison. EPC Information Service - Data Model and Queries. Technical report, Auto-ID Center, October 2003.

[23] R. T. Snodgrass. *The TSQL2 Temporal Query Language*. Kluwer, 1995.

[24] K. Finkenzeller. *RFID Handbook : Fundamentals and Applications in Contactless Smart Cards and Identification*. John Wiley & Sons, 2nd edition, March 2003.

[25] RFID 2004 FORUM Report. http://www.wireless.ucla.edu/techreports2/RFID-2004-Forum.pdf.

[26] C. Floerkemeier, D. Anarkat, T. Osinski, and M. Harrison. PML Core Specification 1.0. Technical report, Auto-ID Center, September 2003.

[27] EPCglobal Object Name Service (ONS) 1.0. Technical report, EPCGlobal Inc, April 2004.

[28] Siemens RF Identification and Locating Systems MOBY. www.siemens.com/rfid.

[29] S. Sarma. Integrating RFID. *ACM Queue*, 2(7):50–57, 2004.

[30] R. Want. The Magic of RFID. *ACM Queue*, 2(7):40–48, 2004.

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

[32] M. Harrison, J. Brusey, H. Moran, and D. McFarlane. PML Server Developments. Technical report, Auto-ID Center, October 2003.

[33] Epcglobal. http://www.epcglobalinc.org.

[34] Microsoft's RFID 'Momentum' Includes Middleware Platform, Apps. http://www.eweek.com/ article2/0,1759,1766050,00.asp, Febrary 2005.

[35] UCLA WinRFID Middleware. http://www.wireless.ucla.edu/rfid/winrfid/.

[36] P.Bonnet, J. Gehrke, and P. Seshadri. Towards Sensor Database Systems. In *MDM*, pages 3–14, 2001.

[37] S. R. Madden. *The Design and Evaluation of a Query Processing Architecture for Sensor Networks*. PhD thesis, University of California, Berkeley, 2003.

[38] P.P.-S. Chen. The Entity-Relationship Model - Towards a Unified View of Data. *ACM Trans. on Database Systems*, 1(1):9–36, 1976.

[39] H. Gregersen and C. S. Jensen. Temporal Entity-Relationship Models - a Survey. *IEEE Trans. on Knowledge and Data Engineering*, 11(3):464–497, 1999.