# A Linear Algebra Approach for Supply Chain Management*

Roberto De Virgilio and Franco Milicchio

Dipartimento di Informatica e Automazione
Università Roma Tre, Rome, Italy
{dvr,milicchio}@dia.uniroma3.it

**Abstract.** In current trends of consumer products market, the amount of RFID data in supply chain management is vast, posing significant challenges for attaining acceptable performance on their analysis. Current approaches provide hard-coded solutions, with high consumption of resources; moreover, these exhibit very limited flexibility dealing with multidimensional queries, at various levels of granularity and complexity. In this paper we propose a general model for supply chain management based on the first principles of linear algebra, in particular on *tensorial calculus*. Leveraging our abstract algebraic framework, our technique allows both quick decentralized on-line processing, and centralized off-line massive business logic analysis, according to needs and requirements of supply chain actors. Experimental results show that our approach, utilizing recent linear algebra techniques can process analysis efficiently.

## 1 Introduction

In the management of a supply chain, main retailers are investing in new technologies in order to boost the information exchange. To this aim, RFID, the Radio-Frequency Identification, is a recent potential wireless technology. An RFID application usually generates a stream of tuples, usually called *raw data*, of the form of a triple $(e, l, t)$, where $e$ is an EPC, $l$ represents the location where an RFID reader has scanned the $e$ object, and $t$ is the time when the reading took place. Other properties can be retrieved, *e.g.*, temperature, pressure, and humidity. A single tag may have multiple readings at the same location, thus potentially generating an immense amount of raw data. Therefore, a simple data cleaning technique consists in converting raw data in *stay records* of the form $(e, l, t_i, t_o)$, where $t_i$ and $t_o$ are the time when an object enters or leaves a location $l$, respectively. In this manuscript, we address the challenging problem of efficiently managing the tera-scale amount of data per day, generated by RFID applications (cf. [1], and [2,6,7]), focusing on stay records as the basic block to store RFID data.

* This is the extended abstract of a paper that appears in the Proceedings of CIKM'11, October 24–28, 2011, Glasgow, Scotland, UK.

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6, 

***Contribution.*** Leveraging such background, this paper proposes a general model of supply chains, mirrored with a formal *tensor* representation (*i.e.* a generalization of linear forms, usually represented by matrices) and endowed with specific operators, allowing both quick decentralized on-line processing, and centralized off-line massive business logic analysis, according to needs and requirements of supply chain actors.

***Outline.*** Our manuscript is organized as follows. In Section 2 we will briefly recall the available literature. The general supply chain model, accompanied by a formal tensorial representation is supplied in Section 3, subsequently put into practice in Section 4, where we provide the reader a method of analyzing RFID data within our framework. We benchmark our approach with several test beds, in Section 5, while Section 6 sketches conclusion and future work.

## 2 Related Work

In a real scenario, great lapse and huge amounts of data are generated. To this aim, knowledge representation techniques focus on operating deep analysis in such systems. An alternative approach [6] consists in warehousing RFID data and performing multidimensional analyses on the warehouse. The focus here is on data compression techniques and on storage models with the goal of achieving a more expressive and effective representation of RFID data. A straightforward method is to provide a support to path queries (*e.g.*, find the average time for products to go from factories to stores in Seattle) by collecting RFID tag movements along the supply chain. Usually, the tag identifier, the location and the time of each RFID reading is gathered and stored in a huge relational table. Lee et al. have proposed an effective path encoding approach to represent the data flow representing the movements of products [7]. In a path, a prime number is assigned to each node and a path is encoded as the product of the number associated with nodes. Mathematical properties of prime numbers guarantee the efficient access to paths. A major limitation of the majority of these approaches have to fix the dimensions of analysis in advance to exploit *ad-hoc* data structures to be maintained. It follows that, in many application scenario the compression loses its effectiveness and the size of tables does not be reduced significantly. Therefore we have a limited flexibility when multidimensional queries, at varying levels of granularity and complexity, need to be performed.

## 3 RFID data Modeling

This section is devoted to the definition of a general model capable of representing all aspects of a given supply chain. Our overall objective is to give a rigorous definition of a supply chain, along with few significant properties, and show how such representation is mapped within a standard tensorial framework.

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6, 

### 3.1 A General Model

Let us define the set $\mathcal{E}$ as the set of all EPCs, with $\mathcal{E}$ being finite. A *property* of an EPC is defined as an application $\pi : \mathcal{E} \to \Pi$, where $\Pi$ represents a suitable property codomain. Therefore, we define the application of a property $\pi(e) := \langle \pi, e \rangle$, *i.e.*, a property related to an EPC $e \in \mathcal{E}$ is defined by means of the pairing EPC-property; a property is a surjective mapping between an EPC and its corresponding property value. A *supply chain* is defined as the product set of all EPCs, and all the associated properties. Formally, let us introduce the family of properties $\pi_i$, $i = 1, \ldots, k + d < \infty$, and their corresponding sets $\Pi_i$; we may therefore model a supply chain as the product set

$$\mathcal{S} = \mathcal{E} \times \Pi_1 \times \ldots \times \Pi_{k-1} \times \Pi_k \times \ldots \times \Pi_{k+d} \,. \tag{1}$$

### 3.2 Properties

In the following we will focus on some of the properties related to EPCs, *i.e.*, we will model some codomains $\Pi$ and their associated features.

***Location.*** Let us briefly model the *location* associated to an EPC. It is common to employ a GPS system in order to track the position on earth, however, any fine-grained space is sufficient to our purposes. In particular, being the earth homeomorphic to a 3-sphere, any ordered triple of real numbers suffices, leading us to the following definition:

**Definition 1 (Location)** *Let $\mathcal{L}$ be the set of ordered tuples $\ell := (\ell_1, \ell_2, \ell_3)$, with $\ell_1, \ell_2, \ell_3 \in \mathbb{R}$. We name $\mathcal{L}$ as* location set*, $\ell_1$, $\ell_2$ and $\ell_3$ as* location coordinates*.*

***Time.*** In order to model a temporal interval relative to a product (EPC), we resort to an ordered couple of elements from the ring of real numbers. This suffices to specify, *e.g.*, the *entry* and *exit* time of a product from a given location. With this point of view, we model time as follows:

**Definition 2 (Time)** *Let $\mathcal{T}$ be the set of ordered couples $\tau := (t_i, t_o)$, with $t_i, t_o \in \mathbb{R}$. We name $\mathcal{T}$ as* time set*, $t_i$ and $t_o$ as* incoming *and* outcoming timestamps*, respectively.*

**Definition 3 (Inner sum)** *Let us define the* inner sum *of two time elements $\tau_1 = (t_i^1, t_o^1)$, $\tau_2 = (t_i^2, t_o^2)$ as the operator $\oplus : \mathcal{T} \times \mathcal{T} \to \mathcal{T}$:*

$$\tau_1 \oplus \tau_2 := \left( \min(t_i^1,\, t_i^2), \max(t_o^1,\, t_o^2) \right) \,.$$

With the above operator, we have that $(\mathcal{T}, \oplus)$ assumes the algebraic structure of an abelian group. Such operation allows us to rigorously model the "addition of products": the overall timeframe of two products is, in fact,

**Definition 4 (Lifetime)** *We define as* lifetime *the linear form $\lambda : \mathcal{T} \to \mathbb{R}$ defined as follows: $\langle \lambda,\, \tau \rangle := t_o - t_i \,, \quad \tau = (t_i, t_o) \in \mathcal{T}$.*

Due to the linearity, we are allowed to construct equivalence classes in $\mathcal{T}$ as

$$[\widetilde{\tau}] := \{ \tau \in \mathcal{T} : \langle \lambda,\, \tau \rangle = \langle \lambda,\, \widetilde{\tau} \rangle \} \,. \tag{2}$$

The *canonical representative elements* of the above equivalence classes are defined as $(0, t_o)$, with $t_o \in \mathbb{R}$.

**Definition 5 (Admissible time)** *Given a time element $\tau \in \mathcal{T}$, we say that $\tau = (t_i, t_o)$ is* admissible *iff $\langle \lambda,\, \tau \rangle > 0$, with $t_i, t_o > 0$.*

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6, 

### 3.3 Tensorial Representation

Let us now introduce a formal *tensorial framework* capable of grasping all properties related to a supply chain, as proposed in Section 3.1. We divide properties into two categories, *countable* and *uncountable* spaces. This separation allows us to represent countable spaces with natural numbers, therefore mapping their product space to $\mathbb{N}^k$, while leaving the product space of all uncountable properties into a collective space $\mathbb{U}$:

$$\mathcal{S} = \underbrace{\mathcal{E} \times \Pi_1 \times \ldots \times \Pi_{k-1}}_{\mathbb{N}^k} \times \underbrace{\Pi_k \times \ldots \times \Pi_{k+d}}_{\mathbb{U}} . \tag{3}$$

Such mapping will therefore introduce a family of injective functions called *indexes*, defined as: $\mathrm{idx}_i : \Pi_i \longrightarrow \mathbb{N}$ with $i = 1, \ldots, k-1$. When considering the set $\mathcal{E}$, we additionally define a supplemental index, the *EPC index function* $\mathrm{idx}_0 : \mathcal{E} \to \mathbb{N}$, consequently completing the map of all countable sets of a supply chain $\mathcal{S}$ to natural numbers.

**Definition 6 (Tensorial Representation)** *The* tensorial representation *of a supply chain $\mathcal{S}$, as introduced in equation* (1)*, with countability mapping as in* (3) *is a multilinear form $\Sigma : \mathbb{N}^k \longrightarrow \mathbb{U}$.*

A supply chain can be therefore rigorously denoted as a rank-$k$ tensor with values in $\mathbb{U}$, mapping countable to uncountable product space.

### 3.4 Implementation

Preliminary to describing an implementation of our supply chain model, based on tensorial algebra, we pose our attention on the practical nature of a supply chain. Our treatment is general, representing a supply chain with a tensor, *i.e.*, with a multidimensional matrix. Supply chain rarely exhibit completion: practical evidence [5] suggests that products, identified by their EPC, for example, seldom present themselves in every location. The same consideration applies also to other properties, in particular, to countable properties. Hence, our matrix effectively requires to store only the information regarding connected nodes in the graph: as a consequence, we are considering sparse matrices [3], *i.e.*, matrices storing only non-zero elements.

*Example 1.* Let us consider the supply chain pictured in Fig. 1, described tensorially by $\Sigma : \mathbb{N}^2 \to \mathcal{T}$, whose representative matrix is as follows:

$$\begin{pmatrix}
(0,2) & \cdot & \cdot & (5,7) & \cdot & (8,10) & \cdot & \cdot \\
\cdot & \cdot & (0,3) & \cdot & \cdot & \cdot & \cdot & (4,9) \\
(0,1) & \cdot & \cdot & (4,5) & \cdot & \cdot & (6,9) & \cdot \\
\cdot & (0,4) & \cdot & \cdot & (5,6) & \cdot & \cdot & (8,9) \\
\cdot & \cdot & (0,5) & \cdot & (6,7) & \cdot & \cdot & (9,11)
\end{pmatrix}$$

where, for typographical simplicity, we omitted $0_{\mathcal{T}} = (0,0)$, denoted with a dot. For clarity's sake, we outline the fact that $\Sigma$ is a rank-2 tensor with dimensions 5 (i.e. the rows), 8 (i.e. the columns). In fact, we have

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6, Copyright (c) 2012 - Edizioni Libreria Progetto and the authors

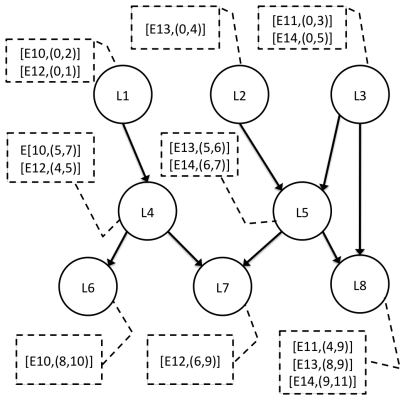


**Fig. 1.** An example supply chain represented by a directed graph. Callouts represent lists of tuple constituted by an EPC with the associated time $(t_i, t_o)$.

$$\mathcal{E} = \{E10, E11, E12, E13, E14\} \ ,$$
$$\mathit{\Pi}_1 = \{L1, L2, L3, L4, L5, L6, L7, L8\} \ ,$$

where $\mathrm{idx}_0(E10) = 1$, $\mathrm{idx}_0(E11) = 2, \ldots, \mathrm{idx}_0(E14) = 5$, and similarly $\mathrm{idx}_1(L1) = 1, \ldots, \mathrm{idx}_1(L8) = 8$. In the sparse representation we have $\{\{1,1\} \to (0,2), \{1,4\} \to (5,7), \ldots, \{5,8\} \to (9,11)\}$.

## 4 RFID data Analysis

Based on our conceptual framework, in this section we will provide a method to analyze RFID data represented by a tensor.

***Tracking Query.*** A *tracking query* finds the movement history for a given tag identifier $e \in \mathcal{E}$. We can comfortably perform the query efficiently, using the model described in Section 3, by applying the tensor application. Therefore, given $i = \mathrm{idx}(e)$, we build a Kroneker vector as a vector $\delta_i$, with $|\delta_i| = |\mathcal{E}|$, and finally apply of the rank-2 tensor represented by $M_{ij}$ to $\delta_i$: $r = M_{ij}\delta_i$ . For instance, referring to the example pictured in Fig. 1, let us consider the tag $E13$, we have $i = \mathrm{idx}(E13) = 4$, and therefore our vector will be $\delta_4 = \{\{4\} \to 1\}$. Consequently, the resulting vector will be $r = M_{ij}\delta_4 = \{\{2\} \to (0,4),\ \{5\} \to (5,6),\ \{8\} \to (8,9)\}$ or in another notation, $L2 \to L5 \to L8$.

***Path Oriented Query.*** A *path oriented query* returns the set of tag identifiers that satisfy different conditions. Following the query templates given in [7], we subdivide path oriented queries into two main categories: *path oriented retrieval* and *path oriented aggregate* queries. The former returns all tags covering a path satisfying given conditions, while the latter computes an aggregate value. It is possible to formulate a grammar for these queries, similar to XPath expressions; in particular, path oriented retrieval and path oriented aggregate queries are respectively indicated as follows: $L_1[\mathrm{cond}_1]//L_2[\mathrm{cond}_2]//\ldots//L_n[\mathrm{cond}_n]$ or

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6, 

$L_1[\text{cond}_1]/L_2[\text{cond}_2]/\ldots/L_n[\text{cond}_n]$, where $L_1, \ldots L_n$ is the *path condition*, *i.e.*, the sequence of locations covered by the tag, and $\text{cond}_i$ is the *info condition*; for more information about such expressions, we refer the reader to [7]. A path condition expresses *parenthood* between locations, indicated as $L_i/L_j$, or *ancestry* with $L_i//L_j$; an info condition, on the other hand, indicates conditions on tag properties, *e.g.*, $StartTime$ and $EndTime$.

In our framework, a *path oriented retrieval query* is easily performed by exploiting the tensor application coupled with the Hadamard product. Given the location set $\mathcal{L}$, for each $z = \text{idx}(L_k)$, with $k = 1, \ldots, n$, we create a Kroneker vector $\delta_z$ and subsequently apply the rank-2 tensor, resulting in a set of vectors $r_z = M_{ij}\delta_j$, where for typographical reasons, we dropped the subscript intending $\delta \equiv \delta_z$. Finally, we apply the condition function to each $r_z$ employing the map operator. This yields a set of vectors

$$\widetilde{r_z} = \text{map}(\text{cond}_z, r_z)\,, \qquad \text{cond}_z : \mathbb{N}^k \times \mathbb{U} \longrightarrow \mathbb{F}\,,$$

whose Hadamard multiplication generates the final result: $\bar{r} = \widetilde{r_1} \circ \ldots \circ \widetilde{r_n}$, reminding the reader that only non-zero values are stored, and therefore given as a result of a computation. The cond functions, as indicated above, are maps between properties of supply chains and a suitable space $\mathbb{F}$, *e.g.*, natural numbers for a boolean result. For an example, referring to Fig. 1, let us consider the query

$$L3[StartTime > 0]//L8[EndTime - StartTime < 4]\,.$$

In this case, given $\text{idx}(L3) = 3$ and $\text{idx}(L8) = 8$, we build $\delta_3$ and $\delta_8$, and generate the partial results : $r_3 = M_{ij}\delta_3 = \{\{2\} \to (0, 3), \{5\} \to (0, 5)\}$ and $r_8 = M_{ij}\delta_8 = \{\{2\} \to (4, 9), \{4\} \to (8, 9), \{5\} \to (9, 11)\}$, where evidently the application was performed along the second dimension, *i.e.*, for each $\delta \in \{\delta_3, \delta_8\}$, we compute $M_{ij}\delta_j$. Finally, subsequent to mapping conditions on the results, in this case $\text{cond}_3 = t_i(\cdot) > 0$ and $\text{cond}_8 = \langle \lambda, \cdot \rangle < 4$, we obtain the correct outcome $\bar{r} = \widetilde{r_3} \circ \widetilde{r_8} = \{\{5\} \to (9, 11)\}$, *i.e.*, $E14$.

Considering parenthood instead of ancestry, *i.e.*, $L_i/L_j$, we briefly sketch the fact that such query does not, in fact, differ from the above, except for one particular: each resulting EPC, when subject to a tracking query, produces a sparse vector whose *length*, *i.e.*, the number of non-zero stored elements, is exactly equal to the number of locations under analysis.

*Path oriented aggregate queries* may be represented as $\langle f, Q \rangle =: f(Q)$ where $f$ is an *aggregate function*, *e.g.*, average or minimum, and $Q$ is the result of a path oriented retrieval query. Therefore, let $r_{\bar{Q}}$ be the result of $Q$, and let $f$ be a function defined on vectors of supply chain elements, we simply have that a path aggregate may be expressed as $f(r_{\bar{Q}})$. Referring to Fig. 1, let us consider the query expressed in the grammar of [7]: $\langle AVG[L8.StartTime], //L8 \rangle$. It is easily performed by applying the function $f := \text{average}(t_i)$ to the outcomes of the path oriented retrieval query on $L8$, resulting in $\bar{r} = \text{average}(\{4, 8, 9\}) = 7$.

## 5 Experiments

We performed a series of experiments aimed at evaluating the performance of our approach, reporting the main results in the present section.

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6, 

***Environment.*** Our benchmarking system is a dual core 2.66 GHz Intel with 2 GB of main memory running on Linux, where we implemented our framework[1] in C++ within the Mathematica 8.0 computational environment. Our results have been compared to the ones from the approach in [7], tested against a generated synthetic RFID data in terms of stay records, and considering products moving together in small groups or individually, a behavior called *IData* in [7]: data production followed the same guidelines on a supply chain of 100 locations. Finally the complete data set comprises $10^5$, $5 \cdot 10^5$, $10^6$, $5 \cdot 10^6$, and $10^7$ stay records. In the following, we will denote our tensorial approach $T$, while the proposed one in [7] with $P$.

***Results.*** Performances have been measured with respect to data loading and querying. Referring to the former, the main advantage of our approach is that we are able to perform loading without any particular relational schema, when compared to $P$, where a schema coupled with appropriate indexes have to be maintained by the system. In this case, loading execution times are 0.9, 11, and 113 seconds, for sets of $10^5$, $10^6$, and $10^7$ stay records, respectively; on the contrary, $P$ timings were in order of minutes and hours. Another significant advantage of $T$ relies in memory consumption: we need 13, 184, and 1450 MB to import the above mentioned sets; as a side-note, we highlight the fact that the $10^7$ set required a division in smaller blocks, *e.g.*, $10^6$, due to the limited memory at disposal. With respect to query execution, $T$ presents a similar behavior and advantages with respect to $P$, for both time and memory consumption. We performed *cold-cache* experiments, *i.e.*, dropping all file-system caches before restarting the systems and running the queries, and repeated all the tests three times, reporting the average execution time. As in [7], we formulated 12 queries to test the two systems as reported in [4]. In brief, Q1 is a tracking query, Q2 to Q5 are path oriented retrieval queries, while Q6 to Q12 are path oriented aggregate queries. Due to the nature of $P$, we were able to perform a comparison only on centralized off-line massive analysis. A significant result is the *speed-up* between the two approaches, as shown in Fig. 2.(a); again, Q1 label is dropped due to typographical causes. We computed the speed-up for all data sets as the ratio between the execution time of $P$, and that of our approach $T$, or briefly $S = {}^{t_P}/_{t_T}$. In general, $T$ performs very well with respect to $P$ in any dataset, particularly for queries related to the object transition, *e.g.*, Q2, Q4, Q5, Q10. The query performance of $T$ is on the average 19 times better than that of $P$, 150 times on the maximum, *i.e.*, Q10. Another strong point of $T$ is a very low consumption of memory, due to the *sparse matrix* representation of tensors and vectors. Fig. 2.(b) illustrates the main memory consumption of each query with respect to $10^5$, $10^6$, and $10^7$ stay records. On the average, tracking queries require very few bytes of memory for any dataset, path oriented queries few KBytes, topping 1 MB for $10^7$. Results demonstrate how our approach can be used in a wide range of applications, where devices with limited calculus resources may process large amount of data in an efficient and effective way.

[1] A prototype implementation is available at http://pamir.dia.uniroma3.it:8080/SimpleWebMathematica

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6, 

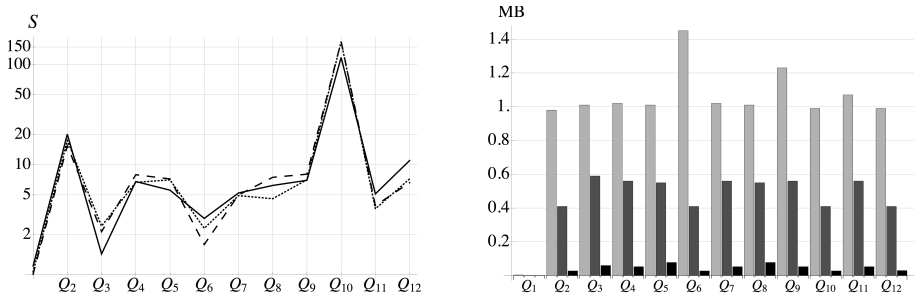


**Fig. 2.** (a) Speed-up logarithmic graph for all queries with $10^5$ (solid), $10^6$ (dashed), and $10^7$ (dotted). (b) Main memory consumption for each query: black bars refer to $10^5$, dark gray to $10^6$, and light gray to $10^7$ data size.

## 6 Conclusion and Future Work

We have presented an abstract algebraic framework for the efficient and effective analysis of RFID data in supply chain management. Our approach leverages tensorial calculus, proposing a general model that exhibits a great flexibility with multidimensional queries, at diverse granularity and complexity levels. Experimental results proved our method efficient when compared to recent approaches, yielding the requested outcomes in memory constrained architectures. For future developments we are investigating the introduction of reasoning capabilities, along with a thorough deployment in highly distributed Grid environments. In addition, we are about to test our model on mobile devices, comprising more complex properties and queries.

## References

1. Angeles, R.: Rfid technologies: supply-chain applications and implementation issues. Information Systems Management 22(1), 51–65 (2005)
2. Bai, Y., Wang, F., Liu, P., Zaniolo, C., Liu, S.: Rfid data processing with a data stream query language. In: ICDE (2007)
3. Davis, T.A.: Direct Methods for Sparse Linear Systems. SIAM (2006)
4. De Virgilio, R., Milicchio, F.: Tensor calculus for rfid data management. Tech. Rep. RT-DIA-182, University Roma Tre (2011)
5. Derakhshan, R., Orlowska, M.E., Li, X.: Rfid data management: Challenges and opportunities. In: RFID. pp. 175–182 (2007)
6. Gonzalez, H., Han, J., Li, X., Klabjan, D.: Warehousing and analyzing massive rfid data sets. In: ICDE (2006)
7. Lee, C.H., Chung, C.W.: Efficient storage scheme and query processing for supply chain management using rfid. In: SIGMOD. pp. 291–302 (2008)

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6, 