DOI: 10.1145/1592761.1592793

BY C. BOLCHINI, C. A. CURINO, G. ORSI, E. QUINTARELLI, R. ROSSATO, F. A. SCHREIBER, AND L. TANCA

And What Can Context Do For Data?

COMMON TO ALL ACTORS IN TODAY'S INFORMATION WORLD is the problem of lowering the "information noise," both reducing the amount of data to be stored and accessed, and enhancing the "precision" according to which the available data fit the application requirements. Thus, fitting data to the application needs is tantamount to fitting a dress to a person, and will be referred to as data tailoring. The context will be our scissors to tailor data, possibly assembled and integrated from many data sources.

Since the 1980s, many organizations have evolved to comply with the market needs in terms of flexibility, effective customer relationship management, supply chain optimization and so on and so forth: the situation where a set of partners re-engineered their single organizations, generating a unique, extended enterprise, has frequently been observed. Together with the organizations, also their information systems evolved, embracing new technologies like XML and ontologies, used in ERP[a] systems and Web-service based applications. In recent years many organizations introduced into their information systems also Knowledge Management features, to allow easy information sharing among the organizations' members; these new information sources and their content have to be managed together with other – we might say legacy – enterprise data. This growth of information, if not properly controlled, leads to a data overload that may cause confusion rather than knowledge, and dramatically reduce the benefits of a rich information system. However, distinguishing useful information from noise, i.e., from all the information not relevant to the specific application, is not a trivial task; the same piece of information can be considered differently, even by the same user, in different situations, or places – in a single word, in a different context.[4,5]

The notion of context, formerly emerged in various fields of research like psychology and philosophy,[3] is acquiring great importance also in the computer science field. In a commonsense interpretation, the context is perceived as a set of variables that may be of interest for an agent and that influence its actions. The context has often a significant impact on the way humans (or machines) interpret their environment: a change in context causes a transformation in the actor's mental representation of the reality, even when the reality is not changed. The word itself, derived from the Latin *cum* (with or together) and *texere* (to weave), describes a context not just as a profile, but *as an active process dealing with the way humans weave their experience within their whole environment, to give it meaning*.

In the last few years, sophisticated and general context models have been proposed to support context-aware applications. In the following we list the different meanings attributed to the word *context*:

- *Presentation-oriented:* context is perceived as the capability of the system to adapt content presentation to different channels or to different devices. These context-models are often rigid, since they are designed for specific applications and rely on a well known set of presentation variables.
- *Location-oriented:* with this family of context models, it is possible to handle

a Enterprise Resource Planning.

time and space coordinates with high precision.[11]

▸ *User-centered:* these models focus on *what the user is doing*. Here, the context history becomes relevant and some kinds of context reasoning are provided; when available, automatic learning is sometimes used to guess user activity from sensor readings.[9]

▸ *Community-based:* an interesting approach, which considers the context as a set of relevant variables shared by a group of peers. Differently from the previous approaches, the context definition is achieved in a distributed fashion.[8]

The aim of this work is to show the way context can be used to define context-aware data views over large information systems, tailoring only the data that are relevant to a given application use-case to obtain a personalized subset of the available information. The use of context over the data has been introduced in references[1,6,10,12] but it is far from being explored in depth. With respect to the above classification, these systems add a new perspective that we can call *data-tailoring-oriented* since they aim at the reduction of the size of data by means of contextual preferences. Moreover, the tailoring process personalizes the retrieved data, thus enhancing the precision of the tailored information. For a survey on such topic see Bolchini.[1]

Conceptually, within an information system, users' knowledge needs may depend on two different aspects: the application *domain* which represents the reality under examination, and the *working environment*, in other words, the context. Classical data models, at a conceptual or at a logical level, are perfectly suited to represent the former, while context modelling presents different challenges and needs appropriate consideration.

The present generation of SQL based commercial DBMSs does not allow an explicit definition and management of context information, but only offers the *view definition* mechanism to determine the part of data which can be managed by each user. It will be shown that our design methodology exploits these common features of DBMSs to realize context-awareness without requiring any change in the datasource design process nor special functionalities of the DBMS. Figure 1 reports the architecture of a context-aware system, exploiting the proposed methodological approach.

Figure 1.

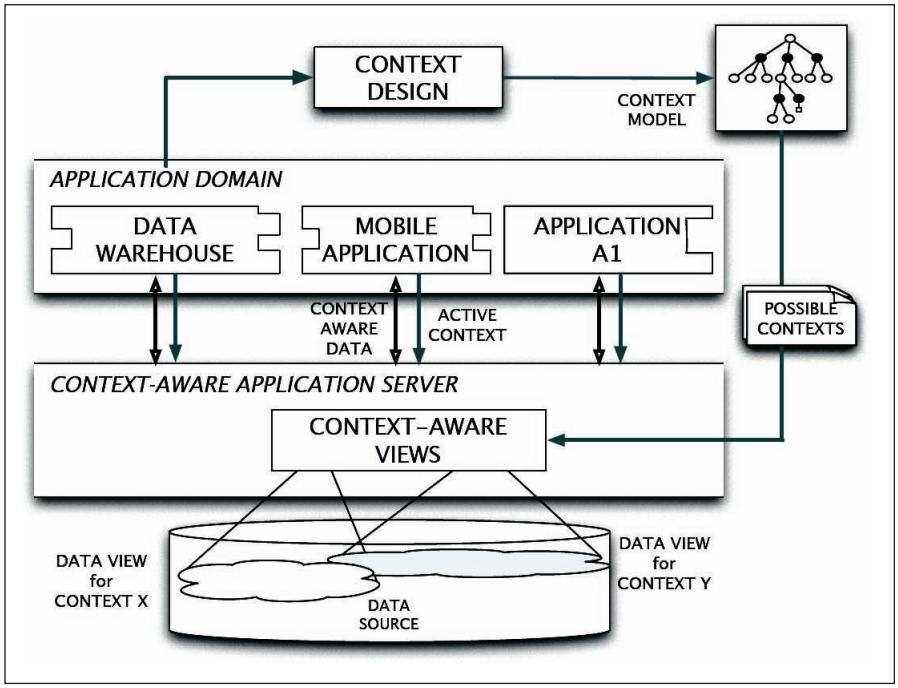


Table 1.

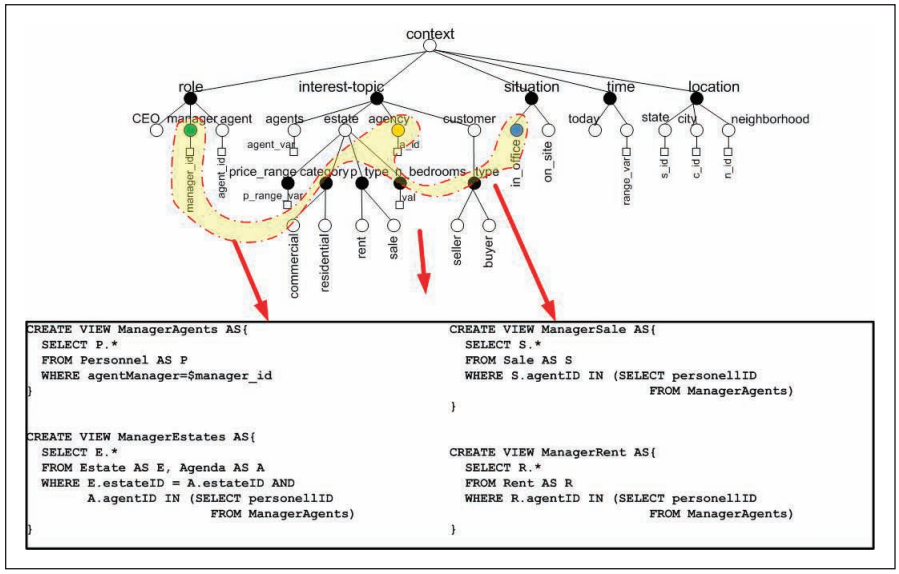
Agency(agencyID, address, operationalCenter, name, openingHours)
Personnel(personnelID, name, surname, birthday, phone, email, address, agentManager, agencyID)
Estate(estateID, ownerID, buildingID, description, squareFeets, cadastralMap)
MultimediaContent(contentID, type, ImageWidth, ImageHeight, mediaData)
EstateMedia(estateID, contentID)
Customer(customerID, cityZip, name, surname, category, address, phoneNumber, email)
CustomerRequest(reqID, buyerID, budget, buyOrRent)
Agenda(date, time, agentID,estateID, custumerID)
Visit(estateID, date, agentID, customerID, visitDuration)
Rent(estateID, customerID, dateIn, agentID, ratePrice, condition, duration)
Sale(estateID, ownerID, buyerID, date,agentID, firstPrice, agreedPrice, conditions, agencyPercentage)

In the context design phase, all the possible scenarios (contexts) of the considered application are identified; then, the designer must determine context-aware views, as the portions of the entire dataset which are relevant for each particular actor in each particular context. We propose the definition of a *context-guided methodology* to support the designer in identifying, for a given application scenario, the contexts and the correspondingly interesting subsets of data. Our methodology is composed of three basic elements: a context model, capturing all the aspects – the so-called dimensions – that allows the implicit representation of the possible application contexts; a strategy for identifying, for each dimension, independently of the others, a relevant portion of the entire data schema – the so-called partial views; and a suite of operators for combining the partial views to derive the final context-aware view(s) associated with each context.

Running Example. Let us consider, as an example, a real estate company, with some operational centers and a country-wide network of franchising agencies; a small portion of the company database schema is reported in Table 1. This company relies on a large information system that is so rich that, sometimes, application developers and end-users are at a loss while formulating queries able to satisfy their necessities. Thus, the company needs a systematic approach to allow each actor to obtain a personalized view over the entire system, without redesigning the information system from scratch.

For example, the chief of an operational center should be able to access only the subset of the entire information system containing agents and

Figure 2.



estates data within his/her center's county, such as estate visits, or signed contracts. In other respects, s/he would also like to have a detailed view of the information related to the portfolio of properties s/he is responsible for.

Moreover, the agents, while in their offices, can access their customers' portfolios, signed agreements, estates assigned to them and their agendas; on the other hand, when on-site for estate visits, they are equipped with smart-phones and, according to the scheduled appointments, only need the information about the features of estates they are going to show, along with the related fragment of the customers' portfolio. Indeed, the presence of a mobile device introduces new challenges, being these devices equipped with reduced computational and storage resources, and subjected to power consumption constraints that limit communication possibilities.

Context modeling. Recently, context has been used as the driving concept for allowing the user a more affordable access to complex sets of Web services;[7] dually, our context model, called Context Dimension Tree,[2] is particularly suited for tailoring the target application data according to the user information needs:[b] discovery, aggregating and tailoring data is indeed somehow complementary to tracking, composition and customization of Web services. Along with the model, we present a design methodology which guides the designer in the data tailoring task.

The Context Dimension Tree (CDT) (see upper part of Figure 2) models context in terms of a set of *context dimensions*, each capturing a different characteristic of the context. A dimension value can be further analyzed with respect to different viewpoints (called sub–dimensions), generating a subtree in its turn. In a CDT, black nodes represent dimensions (such as the interest topic) and sub–dimensions (such as price range, and category); white nodes represent the values the dimensions can assume (such as for the role dimension, CEO, manager, and agent), thus actualizing contexts.

However, in some cases, the number of possible values a (sub–)dimension may assume can be very large (such as, when they are constituted by a range of numerical values) and it would be cumbersome to individually represent each one of them as a white node. Therefore, without modifying the expressive power of the model, *attribute nodes* (square nodes) have been introduced, whose instances are the admissible values for that (sub–)dimension.

Similarly, the need arises to select specific instances in the set of values represented by a white node. In this case, a square node attached to a white node expresses a *restriction parameter* which can be used to single-out data pertaining to the required element.[2] The parameter can be a constant value (for example, "`Westwood`"), a variable name whose value is acquired from the application (such as, `$agent_id`) or the result of a function computation (for example, `getDate()`). In any case, the leaves of the CDT can only be either white nodes or attribute nodes.

In our model a *context* is described by means of *context elements*. A context element may have two different specifications:

dim_name : value or *dim name : value(param value)*

where *dim name* is the name of a (sub-) dimension, and value is a *value* (possibly restricted by a parameter) for that dimension. The following is a possible context from the CDT reported in Figure 2:

```
< role : manager($manager id),
interest topic : agency,
situation : in office >          (1)
```

This is the context of a manager– whose `$manager_id` will be instantiated at run-time when the data are to be retrieved – who is currently at the office, and needs to check information related to agencies he/she is responsible for.

Another possible context is:

```
< role : agent($agent id),
interest topic : estate,
situation : on site,
time : today(getDate),
location : city($c id)) >        (2)
```

It represents the context of an agent – also in this case `$agent id` will be instantiated at run-time – during a site visit. Time and location are automatically derived by the system where data are actually retrieved. Note that the context element `situation:on site` determines the fact that only site-related information is desired since the agent will be out of the office, equipped with a small, mobile device.

The depth of the CDT depends on the granularity used by the designer to specify the various points of view that contribute to the selection of a portion of the entire data set; some dimensions are inherently prone to being detailed, such as the `interest topic`, usually characterized by more than one level, while others have typically a coarse-grained classification of values, for example, the `situation`. The more details in the CDT, the more restrictive the query to derive the final view associ-

b It is not a coincidence that the title of this article directly refers to such previous work.

ated with a context is; it is the designer who shall choose the best trade-off.

According to our experience, some context dimensions are common to most applications. Moreover, not all the listed dimensions are always necessary, while additional ones might be required: the most common dimensions, their meaning and some examples related to our application are reported in Table 2.

Note that privacy and security issues may become relevant in many applications. Thus, an *ownership* dimension can be defined, with a special role: while other dimensions are used to tailor the data, this dimension is used to define access rights – expressed in terms of grant and revoke primitives – to the tailored views. The conditions imposed by the ownership context elements are to be used for 'obscuring' parts of the views derived from the other dimensions. The functionality of this dimension is thus different from that of the others, and we don't discuss it anymore in this paper.

At design time, once the CDT has been defined, the list of its contexts is combinatorially generated. However, given an application scenario and the corresponding CDT, not necessarily all the possible combinations of context elements make sense. The model allows the expression of *constraints* or *preferences* among the values of a context definition to avoid the generation of meaningless ones. As an example, a constraint might indicate that a context where the values CEO and on-site are present at the same time is forbidden, because a CEO will never be out for a visit. Here we do not delve into the use of constraints, which are thoroughly dealt with in Bolchini et al.[2]

Context-aware data tailoring. Given the possible and meaningful contexts, the subsequent work of the designer consists in associating them with the relevant portions of the information domain. This step can take two different directions, one more human-intensive and time-consuming, but leading to a more precise view production, the second one more automatic, but more prone to possible errors, thus to be verified a-posteriori. The two strategies are called *configuration-based mapping* and *value-based mapping, respectively*.

When the *configuration-based mapping* strategy is adopted, the designer manually associates each one of the previously produced contexts with the corresponding portion of the information domain schema. This can be done by defining a view in the language supported by the underlying database, or by selecting these portions by means of a graphical interface which will automatically derive the corresponding view. We developed a tool to support the latter technique, producing views from a graphical, E-R specification.[1]

Let us now consider the context of a manager who is interested to information related to his/her agencies, as in the context definition (1) Suppose the designer associates with it the set of SQL views reported in Figure 2. In particular, s/he deems significant for the specific context, the data related to personnel working in the agencies controlled by that manager, sale and rent contracts, with the corresponding estate information.

This kind of work must be done for all significant contexts, whose number is, in general, very high: it amounts to about 1.400 contexts in our example; which reduce to 864 meaningful contexts after constraint application. Thus the task of associating relevant data views with each of them is highly unpractical. Moreover, if the context model changes, for example, if a new dimension, or even just a context element, is inserted, a relevant number of new contexts will have to be generated from its combination with the other existing context elements, and the designer will have to redefine all the related mappings.

The *value-based mapping* strategy is compositional, thus allows us to overcome the limitations of the configuration-based approach. According to this strategy, the designer manually selects the partial view, such as the portion of the global schema, that is relevant for each single context element, without considering the other ones. Then, the view related to each specific context is obtained by means of an algorithm that automatically combines the partial views of its elements, deriving the view definition that is suitable for the given context.

Figure 3 shows, under the CDT, the partial views of each element of the context defined by (2). The bottom of the figure shows the corresponding views over the real estate relational database. The colored dotted lines represent the associations between each context element and its partial view. The agent will be interested in: details about the properties of the current area, multimedia content compatible with the portable device and associated with the properties (also with the possibility to update it with new pictures), and his/her own agenda of the next few days, in order to be able to arrange visits. The instantiation of the agent, the day and the zone is appropriately fed to the system at run time, as previously discussed.

Table 2.

Dimension	Meaning	Examples
Role	The actors using the system.	`CEO`, `agency manager`, `agent`.
Interest Topic	The areas of interest for the possible users of the application.	"`agencies`", i.e., information about agencies (and one in particular) that can be controlled by the CEO, "`agents`", i.e., information about agents that can be viewed by the CEO or by the agency's manager, "`customers`", i.e., information about sellers and buyers that can be viewed by the agent and manager, and "`properties`", i.e., all the knowledge about estates to be sold or rented. This last interest topic can be further decomposed w.r.t. two different criteria: commercial/residential estates or rented/sold properties.
Situation	Phases of the application life.	The user is consulting his/her data when at the office, i.e., `in_office` as opposed to the `on_site` situation, when, for instance, an agent is showing an apartment to a prospective customer.
Time	Temporal indication based on the current time. Time can be relative or absolute and its granularity may vary.	In our example we have chosen a relative view of time, and allowed two choices: the current day, or a variable time interval centered on the current instant, suitable for data analysis.
Space	A location indication, normally referring to the place where the user is currently located. Space can be relative or absolute, and its granularity may vary.	In our example, "`here`" or "`this city`" are relative space data, while "`Marina del Rey district in L.A.`" is absolute.
Interface	Indication of channel or presentation for delivering information.	In some application cases, some data have to be used by humans, directly perusing text and multimedia information, but in others data could be managed by electronic devices solely, requiring only compact codes.

Figure 3.

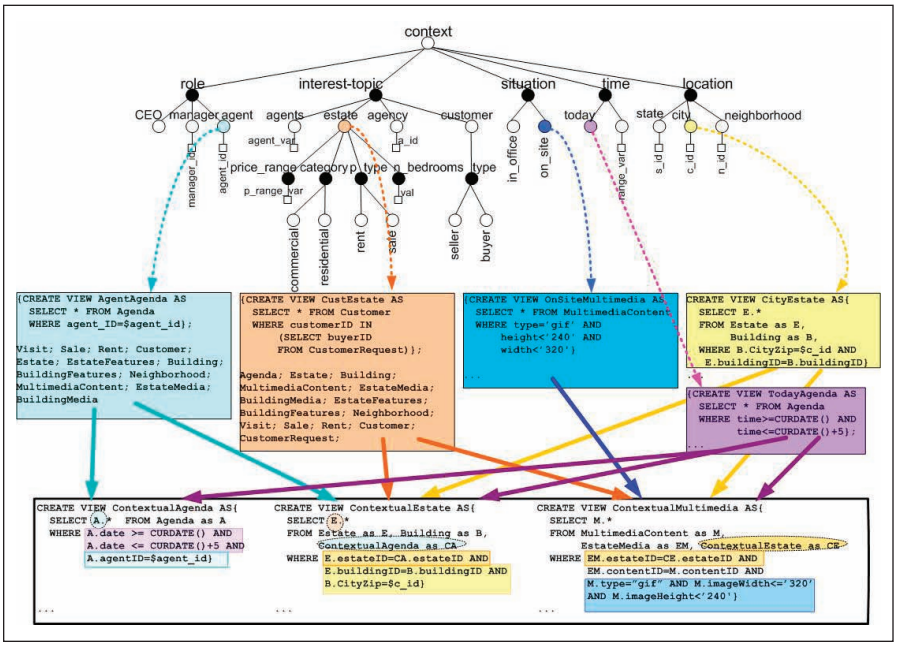


Table3.

Context	Corresponding data
⟨role:agent($agent_id), interest_topic:estate, situation:on_site, time:today, location:city($c_id)⟩	Agent's operational daily data: schedule estates details (including building facilities and neighborhood amenities), multimedia data compatible with his/her mobile device, agent's agenda, estate owner contact, customer contact.
⟨role:CEO, interest_topic:sales, time:month, location:city($c_id)⟩	Aggregated data about sales: e.g., average and sum of sales returns, average of commissions on sales aggregated by time and location.
⟨role:manager($manager_id), interest_topic:agents, situation:in_office, time:range($range_var)⟩	Data to assign the properties to the agents: Agents agenda, personnel and agent data, pending properties, customers requests.
⟨role:agent($agent_id), interest_topic:customers, situation:in_office, location:city($c_id)⟩	Data useful to match buyer and seller: customer's contacts, pending requests and offers details.

Other interesting context examples for our scenario are shown in Table 3, with a brief description of the portion of data they tailor.

Partial view combination can be based on different policies, involving the use of different combination operators.[2] The final view automatically generated from the partial views strictly depends on the granularity used to define partial views and the operators used to combine them. As a consequence, it can be precise or coarse-grained, and may need to be refined a-posteriori by the designer to better suit her/his needs. It is worth noting that the effort for refining automatically defined views is surely lower than that necessary to define each view from scratch.

The combination operators we have defined, called *double union, double intersection* and *double difference*, work on partial views, and extend the algebraic *union*, *intersection* and *difference* operators, respectively, to sets of tables. For example, the final view of Figure 3 has been obtained by applying double intersection to the partial views involved in the considered context (see arrows). The double intersection, when applied to two sets of relations A and B, applies the intersection operator of relational algebra to the greatest common subschema of each pair of relations R_A and R_B (belonging to **A** and **B**, respectively), where either the schema of R_A is a subset of that of R_B or vice-versa. The tables which do not have common sub-schemata do not contribute to the result.

Conclusion and Developments. Up to now we have focused on the use of our context-based methodology to *tailor* huge amounts of possibly noisy data in order to fit actors' needs, but our methodology is also useful to support the design of dimensional information systems, such as data warehouses, which is part of our future work. On the other hand, this work belongs to a wider project, Context-ADDICT, which aims at retrieving and tailoring data coming from disparate and heterogeneous information sources, possibly within a P2P cooperative system, in the frame of an "emergent semantics" approach.

References

1. Bolchini, C., Curino, C. A., Quintarelli, E., Tanca L., and Schreiber, F. A. A data-oriented survey of context models. *SIGMOD Record*, 2007.
2. Bolchini, C., Quintarelli, E., and Rossato, R. Relational data tailoring through view composition. In *Proc. Intl. Conf. on Conceptual Modeling* (ER'2007). LNCS, Nov. 2007.
3. Chalmers, M. A historical view of context. *Computer Supported Cooperative Work 13*, 3, (2004), 223–247.
4. Coutaz, J., Crowley, J. L., Dobson, S., and Garlan, D. Context is key. *Comm. ACM 48*, 3, (Mar. 2005).
5. Dey, A. K. Understanding and using context. *Personal Ubiquitous Computing 5*, 1, (2001), 4–7.
6. Kaenampornpan, M., and O'Neill, E. An integrated context model: Bringing activity to context. In *Proc. Workshop on Advanced Context Modelling, Reasoning and Management*, 2004.
7. Maamar, Z., Benslimane, D., and Narendra, N. C. What can context do for Web Services? *Comm. ACM 49*, 12, (Dec. 2006), 98–103.
8. Ouksel, A. M. In-context peer-to-peer information filtering on the Web. *SIGMOD Record 32*, 3, (2003) 65–70.
9. Petrelli, D., Not, E., Strapparava, C., Stock, O., and Zancanaro, M. Modeling context is like taking pictures. In *Proc. of the Workshop: The What, Who, Where, When, Why and How of Context-Awareness. CHI2000.*
10. Stefanidis, K., Pitoura, E., and Vassiliadis, P. Modeling and storing context-aware preferences. In *Proc. of Advances in Databases and Information Systems, 10th East European Conference*, LNCS, 4152, Springer, 124–140, 2006.
11. Strimpakou, M., Roussaki, I., and Anagnostou, M. E. A context ontology for pervasive service provision. In *20th Int. Conf. on Advanced Information Networking and Applications* (AINA 2006), (2006) 775–779.
12. Yang, S. J. H., Huang, A. F. M., Chen, R., Tseng, S.-S., and Shen, Y.-S. Context model and context acquisition for ubiquitous content access in ULearning environments. In *IEEE Int. Conf. Sensor Networks, Ubiquitous, and Trustworthy Computing*, 2, (2006), 78–83.

This research is partially supported by the Italian MIUR projects: ART-DECO (FIRB), and ESTEEM (PRIN).

Cristiana Bolchini (bolchini@elet.polimi.it) is an associate professor of Computer Engineering at Dipartimento di Elettronica e Informazione of the Politecnico di Milano, Italy.

Carlo A. Curino (curino@elet.polimi.it) is a Ph.D. student at Dipartimento di Elettronica e Informazione of the Politecnico di Milano, Italy.

Giorgio Orsi (orsi@elet.polimi.it) is a Ph.D. student at Dipartimento di Elettronica e Informazione of the Politecnico di Milano, Italy.

Elisa Quintarelli (quintarelli@elet.polimi.it) is a researcher at Dipartimento di Elettronica e Informazione of the Politecnico di Milano, Italy.

Rosalba Rossato (rossato@elet.polimi.it) is a post-doc at Dipartimento di Elettronica e Informazione of the Politecnico di Milano, Italy.

Fabio A. Schreiber (schreibe@elet.polimi.it) is a Full Professor of Data Bases and Technologies for Information Systems at Dipartimento di Elettronica e Informazione of the Politecnico di Milano, Italy.

Letizia Tanca (tanca@elet.polimi.it) is a Full Professor of Data Bases and Technologies for Information Systems at Dipartimento di Elettronica e Informazione of the Politecnico di Milano, Italy