

Using Context for the Extraction of Relational Views^{*}

Cristiana Bolchini, Elisa Quintarelli, Rosalba Rossato, and Letizia Tanca

Dip. Elettronica e Informatica, Politecnico di Milano

P.zza L. da Vinci, 32 - 20133 Milano, Italy

{bolchini, quintare, rossato, tanca}@elet.polimi.it

Abstract. The paper presents an approach for automatically extracting views from a relational database schema, based on the knowledge of the application domain. In order to achieve such result, the possible contexts emerging from the application domain are identified and, starting from the representation of the characterizing dimensions, a sequence of operations is applied to the schema to derive the portion of interest or, as we say, to *tailor the schema* with respect to each possible context. We introduce the tailoring operators as well as the method to compose partial views to obtain the final ones; a running example is used to lead the reader through the steps of the proposed methodology.

1 Introduction

The notion of “context” has received a lot of attention in the last years, due to the perception that context has often a significant impact on the way humans (or machines) act and on how they interpret things; furthermore, a change in context causes a transformation in the experience that is going to be lived. The word itself, derived from the Latin *con* (with or together) and *texere* (to weave), is descriptive; a context is not just a profile, it is an active process dealing with the way we *weave* our experience *together* to give it meaning. In general, the things we see or do remain unchanged, but the perception is different.

When dealing with the reality of *information management*, context plays a significant role in determining what portion of the entire information is relevant with respect to the ambient conditions. Furthermore, effective data access and management support is indispensable both when considering the huge amount of data usually available, which may confuse the user if not opportunely filtered, and when dealing with devices equipped with reduced resources where only the most valuable data should be kept.

Given this scenario, we propose a context-driven approach to extract data views, based on a) the identification of the context dimensions and their values with respect to the current application scenario, b) the definition of a set of operations to be applied to the entire data set in order to determine the portions of data (partial views) related to the context elements, and c) the combination of such partial views to obtain the data view associated with each possible context.

In this paper we focus the attention on relational databases, and on the extraction of relational views over them; yet, the methodology itself has a broader spectrum. The aim

^{*} This work has been partially funded by projects MIUR-PRIN ESTEEM and MIUR-FIRB ARTDECO.

is to provide support to the designer of data management applications, be them related to huge (e. g., in data warehousing) or to a very small amount of data (e. g., in portable, lightweight data management systems), in determining and creating the various views to be used in the different contexts, by following a systematic approach. Such a structured methodology also allows to keep the complexity of the operation at an acceptable level and to facilitate maintenance when new context dimensions or values come into play.

These issues of context definition and context-aware view extractions are nowadays particularly investigated, due to the necessity of rationalizing the access to too much information, in articulated application scenarios. A detailed discussion on related work is presented in Section 5.

The rest of this paper is organized as follows. Section 2 illustrates in detail the goal of this approach, highlighting the main contribution of our work and introducing the adopted case study. Section 3 presents the two strategies for view definition, investigating their benefits and limitations, also with respect to the perception the designer has of the application scenario and data. In Section 4 the methodology is applied to the example scenario, walking the reader through the main phases to the result; related work is discussed in Section 5, whereas concluding remarks and considerations on future work are drawn in the last section.

2 Goals and Contributions

Let us consider the relational database of a large real estate agency that conducts negotiations related to flats, shops and houses located in different areas of the city.

The agency database stores data related to customers, estates, owners, and about its own agents. Moreover, when an agent enters into a contract with a customer, the information about the sale (or rent) contract is stored in the database. Our objective is designing a number of views, to be made available to different possible actors of the scenario: supervisors, agents, buyers and sellers. Supervisors are typically managers who are responsible for the agency business at a high level; supervisors are often the first to be in touch with the sellers, who establish the first contact with the top levels of the agency; subsequently, supervisors assign the sale to one or more agents, who will be in charge for it. The agents are also the first contacts for possible buyers of the estates assigned to them. The agency promotes its sales by various means, among which a Web site, which contains information for possible sellers and buyers, whose views are also to be designed over the corporate database.

The first step of the proposed approach consists in specifying the ambient dimensions characterizing the possible contexts in this application scenario, also listing the expected values, according to the methodology presented in [1].

More precisely, our context model, called *Context Dimension Tree*, is an extension of the context model presented in [2], and is used to systematically describe the user needs, and to capture the context the user is acting in. It plays a fundamental role in tailoring the target application data according to the user information needs. Once the possible contexts have been designed, each must be connected with the corresponding view definition. In this way, once the context becomes current, the view is computed

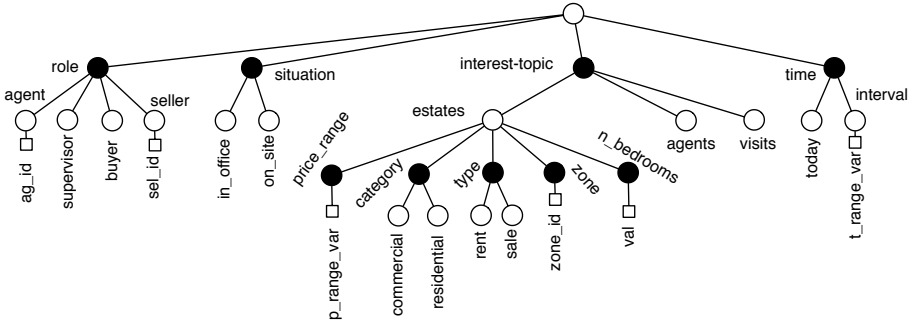


Fig. 1. The Context Dimension Tree

and delivered to the user. The various ways to operate such design-time connection are the subject of this paper.

The tree can be specified in several ways; in particular we propose an ontological representation in the OWL language [3], and an XML representation based on a DTD [4]. In the Context Dimension Tree, the root’s children are the *context dimensions* which capture the different characteristics of the users and of the context they are acting in; by specifying a value for each dimension, a point in the multidimensional space representing the possible contexts is identified, a *context element*. A dimension value can be further analyzed with respect to different viewpoints, generating a subtree in its turn. However deep the tree, the complex structure originating from the composition of the Context Dimension Tree elements is called a *context* or *chunk configuration*, and determines a portion of the entire data set, i.e., a *chunk*, specified at design-time, to be selected later, at run-time, when the corresponding context becomes active. Black nodes represent dimensions and sub-dimensions, while white nodes represent the values they can assume, and more precisely, a context element is a concept node (white node), such as the *supervisor*, or a dimension node with a variable (black node) *price_range* (*p_range_var*), elements that determine a point in the space of all possible contexts.

A collection of one or more chunks constitutes the view the system makes available for the user when he or she is in a particular context. Fig. 1 shows the contexts we have elaborated for our real estate agency. The result is quite intuitive, we refer the interested reader to [5] for a more in-depth discussion on the Context Dimension Tree model.

It is worth noting that, in general, the Context Dimension Tree can be designed independently of the data schema. In the present application scenario, the single agency database is composed by the tables reported in Fig. 2.

Let us consider the particular situation of an agent who is currently at the office, and needs to check the sales of residential estates located in a certain area that have been concluded today. Thus, the current context (or chunk configuration) *C* is composed by the nodes

$$\{\{agent(\$ag_id)\}, \{in_office\}, \{today\}, \{residential, sale, zone(\$zone_id)\}\}$$

allowing the selection of the information related to the interesting sales.

OWNER(IdOwner, Name, Surname, Type, Address,City, PhoneNumber)
ESTATE(IdEstate, IdOwner, Category, Area, City, Province, RoomsNumber,
Bedrooms, Garage, SquareMeters, Sheet, CadastralMap)
CUSTOMER(IdCustomer, Name, Surname, Type, Budget, Address, City, PhoneNum)
AGENT(IdAgent, Name, Surname, Office, Address,City,Phone)
AGENDA(IdAgent, Data, Hour, IdEstate, ClientName)
VISIT(IdEstate, IdAgent, IdCustomer, Date, VisitDuration)
SALES(IdEstate, IdAgent, IdCustomer, Date, AgreePrice, Status)
RENT(IdEstate, IdAgent, IdCustomer, Date, RatePrice, Status, Duration)

Fig. 2. The database schema

Such configuration must be associated at design time with the (definition of the) piece of data which must become available when C occurs, and the same must be done for each possible (and reasonable) chunk configuration. Note that the instantiation of the parameter \$zone_id is fed to the system at run time (e.g. with the value “Piola”). For this purpose, our context model provides *variables*, graphically represented as small squares attached to concept nodes of the Context Dimension Tree, to be suitably used by the run-time system.

Thus, at design time, once the tree has been defined, the list of its chunk configurations is combinatorially derived. Note that not all possible combinations make sense for a given scenario; the model allows the expression of constraints or preferences on the allowed combinations of the dimension values, thus the meaningless ones are not generated. We do not further elaborate on constraints, which are thoroughly dealt with in [5].

The subsequent work of the designer can take two different directions, one more time-consuming but allowing for a more precise view production, the second one more automatic but more prone to possible errors, thus to be verified a-posteriori.

When the first strategy is adopted, once the chunk configurations are combinatorially derived the designer associates each of them with the corresponding portion of the information domain schema. This can be done by directly writing a query in the language supported by the underlying database, or by selecting these portions by means of a graphical interface which will derive the corresponding view. This process is already supported by a tool we have developed [6]. However, as it can be expected, even after excluding the meaningless configurations, a medium-size Context Dimension Tree originates a huge number of chunk configurations, thus the task of associating relevant chunks to each of them is unpractical.

According to the second strategy, the designer must select the schema portion to be associated with *each context element*, and then the system will combine them within each chunk configuration. The possible combination policies involve the use of different combination operators, such as intersection or union, and the designer should choose different association methods according to the chosen operators.

The next section presents the two proposed strategies, discussing benefits and limitations.

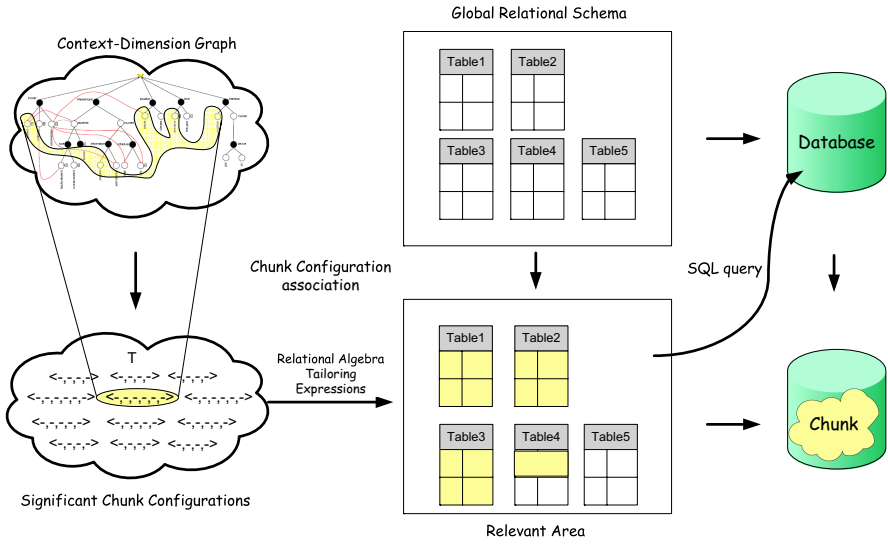


Fig. 3. The Configuration-based mapping strategy

3 Strategies for View Definition

In this section we describe our proposal to define a context-dependent view – expressed as a set of relational algebra expressions – for each chunk configuration. To achieve this goal we have developed two possible strategies: the *configuration-based mapping* and the *value-based mapping*.

Configuration-based mapping

The mapping specifies, for each context (i. e., chunk configuration) C , the set of queries that define the portion of the global database useful for that context. In particular, the designer associates with each chunk configuration the schema chunk (i.e., the view) that s/he deems significant for that specific configuration. Fig. 3 shows the dynamics of the configuration-based mapping strategy.

Example 1. As an example, let us now consider the chunk configuration of the previous section:

$$C = \{\{\text{agent}(\$ag_id)\}, \{\text{in_office}\}, \{\text{today}\}, \{\text{residential, sale, zone}(\$zone_id)\}\}$$

According to the configuration-based mapping, the designer has to select the portion of the global database that is relevant for the current context. In this example, a possible chunk associated with the considered chunk configuration is:

$$\begin{aligned} \mathcal{R}el(C) = & \sigma_{(Category="Residential" \wedge Area=\$zone_id)}(ESTATE) \bowtie \\ & (\sigma_{Date=\$TODAY}(SALE) \bowtie CUSTOMER) \end{aligned} \quad \square$$

This strategy has two main disadvantages: (1) the number of chunk configurations for real applications is quite high (e.g., in the present example there are 816 significant contexts), thus, this strategy requires a lot of effort at design time because the designer has

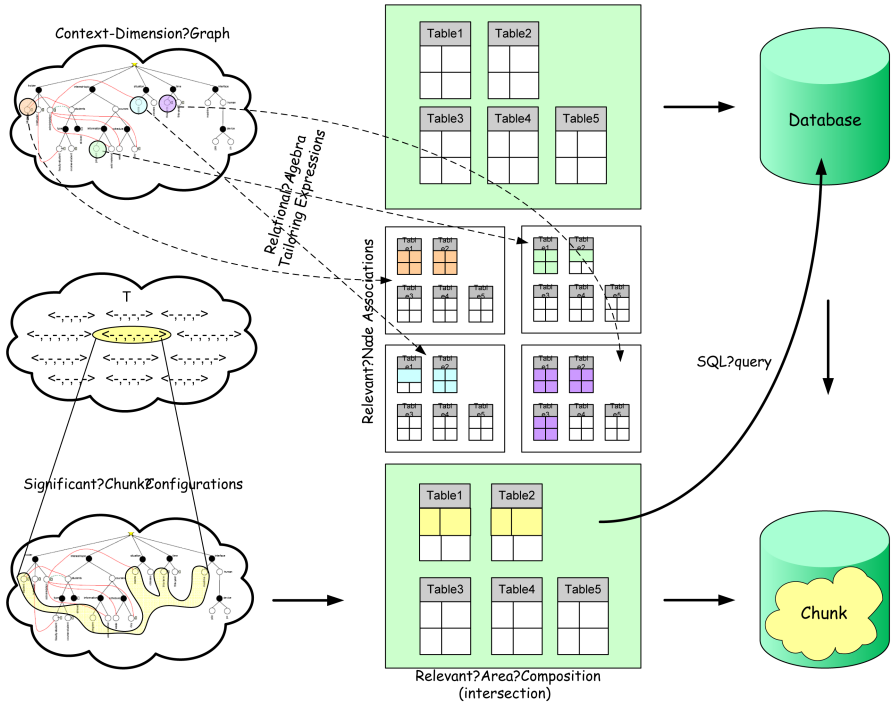


Fig. 4. The Value-based mapping strategy

to manually specify all the views for the chunk configurations resulting from the context model of the target application; (2) if the context model changes, e.g. if a new concept is inserted, a high number of new chunk configurations will have to be generated from its combination with the other concepts, and the designer will have to redefine all the mappings for these.

Value-based mapping

With this strategy, the designer maps each concept node of the tree to a portion (called partial view) of the relational schema. To achieve this goal, he/she labels each node of the Context Dimension Tree with a set of relational algebra expressions that, applied to the global database, allow to define the partial views for the considered node. In the following we refer to this step as *labeling phase*.

After the labeling phase, the view related to each specific chunk configuration is automatically obtained by combining the partial views of its concept nodes. The dynamics of this second strategy is depicted in Fig. 4.

Example 2. For example, the *buyer role* is mapped to a view describing both commercial and residential flats and houses, that is, the *ESTATE* table deprived of the *IdOwner* and *CadastralMap* attributes, which are not interesting for prospective buyers. Formally:

$$\mathcal{R}el(\text{buyer}) = \{\Pi_{Z}ESTATE\}$$

where Z is the set of attributes $\{IdEstate, Category, Area, City, Province, RoomsNumber, Bedrooms, Garage, SquareMeters\}$. The residential *category* is a relational view describing residential flats and houses, that is:

$$\mathcal{R}el(\text{residential}) = \{\sigma_{Category="Residential"}ESTATE\}$$

The view for the $\langle\{\text{buyer}\}, \dots, \{\text{residential}\}\rangle$ chunk configuration is obtained by considering $\mathcal{R}el(\text{buyer})$ and $\mathcal{R}el(\text{residential})$ and applying between them a suitable operator for (partial) view composition. \square

As possible operators to combine relational views, we propose variants of the union and intersection operators of relational algebra; they produce more or less restricted results on the basis of the partial view assigned to each dimension value. More in detail, the final view, i.e. the chunk, associated with a considered context can result more or less restricted on the basis of (i) the granularity used during the labeling process and (ii) the operator used to combine partial views. We propose the following two alternative labeling policies – i.e., policies to define value-based mappings.

1. **Maximal relevant area:** the partial view for a given concept node contains all the schema portions that *might be related* to that concept. For example, for the residential *Category* a possible mapping specified by means of the maximal relevant area policy is:

$$\begin{aligned} \mathcal{R}el(\text{residential}) = \{ & \sigma_{Category="Residential"}ESTATE, \\ & OWNER \times (\sigma_{Category="Residential"}ESTATE), VISIT \times (\sigma_{Category="Residential"}ESTATE), \\ & SALES \times (\sigma_{Category="Residential"}ESTATE), RENT \times (\sigma_{Category="Residential"}ESTATE), \\ & AGENDA \times (\sigma_{Category="Residential"}ESTATE) \} \end{aligned}$$

According to the spirit of the maximal relevant area policy, in this example the view associated with the residential *Category* contains all the information that is related, in some way, to residential properties; that is, besides the ESTATE table with the specific selection conditions, the view also contains all the further information related to residential flats and houses included in the OWNER, VISIT, SALES, RENT, and AGENDA tables.

2. **Minimal relevant area:** the value-based mapping of a given concept node only specifies the data that *are strictly needed* for that concept. For the residential *Category*, the minimal relevant area policy would choose the view: $\mathcal{R}el(\text{residential}) = \{\sigma_{Category="Residential"}ESTATE\}$ which only chooses the tuples of the ESTATE table that refer to residential properties.

It is worth remarking that the hierarchical structure of the Context Dimension Tree has been designed with the idea that lower-level concept nodes correspond, in the data schema, to smaller portions of data, thus the detail level of the data selected while descending the subtree related to each context dimension increases. Thus, it is desirable that the partial view for a node n contain the partial view of each descendant m of n ; i.e. $\mathcal{R}el(n) \supseteq \mathcal{R}el(m)$.

According to our experience, we mean the maximal area policy to assign the widest view for each node of the Context Dimension Tree; consequently, for the designer who

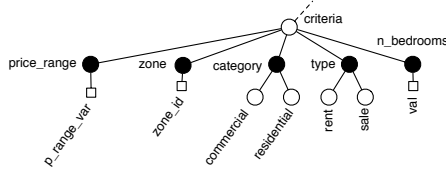


Fig. 5. The portion of the Context Dimension Tree that describes *criteria* information related to estates

applies this policy, it is more natural to perform the labeling phase by navigating the Context Dimension Tree *top-down* (from the root to the leaves). This actually means that the partial view for the root node is the entire database, and, in general, the partial view for any node m is defined by restricting its parent's partial view. Thus, the labeling phase is performed by navigating the Context Dimension Tree top-down and by specifying for each node m a view ($\mathcal{R}el(m)$) over the (previously defined) ($\mathcal{R}el(n)$) of its parent n .

Example 3. Let us now focus our attention on the subpart of the Context Dimension Tree shown in Fig. 5. Consider the maximal area policy; the partial view for the *criteria interest-topic* is a view containing all the information related to flats and houses. Formally:

$$\mathcal{R}el(criteria) = \{ESTATE, OWNER, VISIT, SALES, RENT, AGENDA\}$$

The partial view for the *residential category* (that is the same one reported for the maximal relevant area policy) is a set of views restricting $\mathcal{R}el(criteria)$ only to information about the residential estates:

$$\begin{aligned} \mathcal{R}el(residential) = & \{ \sigma_{Category='Residential'} ESTATE, OWNER \times (\sigma_{Category='Residential'} ESTATE) \\ & VISIT \times (\sigma_{Category='Residential'} ESTATE), SALES \times (\sigma_{Category='Residential'} ESTATE), \\ & RENT \times (\sigma_{Category='Residential'} ESTATE), AGENDA \times (\sigma_{Category='Residential'} ESTATE) \} \quad \square \end{aligned}$$

By contrast, the minimal area policy should be used when the designer prefers to assign the minimal partial chunk for each node of the Context Dimension Tree; in this case, to be adherent with the spirit of the Context Dimension Tree, it is recommended, when this policy has been chosen by the designer, to perform the labeling phase by navigating the Context Dimension Tree *bottom-up*. This means that the mappings are defined starting from the leaf nodes and the view of a non-leaf node can be obtained by composing the views associated with its children. The composition can be realized by using the union operator and possibly adding additional portions of the global database that the designer considers useful for the more general (non-leaf) concept. For example, when using the minimal policy, $\mathcal{R}el(criteria)$ can be obtained by combining $\mathcal{R}el(price_range)$, $\mathcal{R}el(zone)$, $\mathcal{R}el(category)$, $\mathcal{R}el(type)$, and $\mathcal{R}el(n_bedrooms)$.

Since the choice, on the designer's part, of a different labeling policy gives rise to different kinds of mappings, also different operators should be used in the two cases to combine the partial views. These operators can be applied for two purposes: (i) to

derive the chunk associated with a chunk configuration, starting from the partial views, and (ii) to define the partial view associated with non-leaf nodes, as discussed for the bottom-up navigation of the Context Dimension Tree during the labeling phase. The operators and their use will be introduced in the following.

Combination Operators

The context-defined views on a relational database may be written by using relational algebra or, equivalently, in SQL, as shown in Section 4. In order to compose different views, and automatically obtain the view for a given context, we need to introduce a suite of new algebraic operators which work on *sets of relations*, which have been formally defined in [7], and whose intuition is provided in the following.

Let \mathcal{A} and \mathcal{B} be two sets of relations, and $\mathcal{S}_{\mathcal{A}}$ and $\mathcal{S}_{\mathcal{B}}$ the corresponding sets of relational schemata.

Double Union operator: the double union operator $\mathcal{A} \uplus \mathcal{B}$, of \mathcal{A} and \mathcal{B} , returns the union of the set of all relations R_i whose schema is in $(\mathcal{S}_{\mathcal{A}} - \mathcal{S}_{\mathcal{B}}) \cup (\mathcal{S}_{\mathcal{B}} - \mathcal{S}_{\mathcal{A}})$ and a sort of intersection between R_A and R_B , for each pair of relations $R_A \in \mathcal{A}$ and $R_B \in \mathcal{B}$ having the same schema¹.

Example 4. Assume the labeling of the *residential category* and the *buyer role* reported in the Example 2 (minimal area policy). The view associated with the chunk configuration $C = \langle \{\text{buyer}\}, _ , _ , \{\text{residential}\} \rangle$ obtained by applying the Double Union operator between $\mathcal{R}el(\text{buyer})$ and $\mathcal{R}el(\text{residential})$ is the following:

$$\mathcal{R}el(C) = \mathcal{R}el(\text{buyer}) \uplus \mathcal{R}el(\text{residential}) = \sigma_{\text{Category}=\text{"Residential"}}(\Pi_Z(\text{ESTATE}))$$

where Z is the set of attributes $\{\text{IdEstate}, \text{Category}, \text{Area}, \text{City}, \text{Province}, \text{RoomsNumber}, \text{Bedrooms}, \text{Garage}, \text{SquareMeters}\}$. \square

Example 5. Assume the same relevant area for the *residential category*, and for the *agent role* the minimal relevant area containing his/her data and agenda: $\mathcal{R}el(\text{agent}) = \{\sigma_{\text{IdAgent}=\$ag_id}(\text{AGENT}), \sigma_{\text{IdAgent}=\$ag_id}(\text{AGENDA})\}$. The view associated with the chunk configuration $C = \langle \{\text{agent}(\$ag_id)\}, _ , _ , \{\text{residential}\} \rangle$ obtained by applying the Double Union operator between $\mathcal{R}el(\text{agent}(\$ag_id))$ and $\mathcal{R}el(\text{residential})$ is:

$$\begin{aligned} \mathcal{R}el(C) &= \mathcal{R}el(\text{agent}(\$ag_id)) \uplus \mathcal{R}el(\text{residential}) \\ &= \{\sigma_{\text{IdAgent}=\$ag_id}(\text{AGENT}), \sigma_{\text{IdAgent}=\$ag_id}(\text{AGENDA}), \sigma_{\text{Category}=\text{"Residential"}}(\text{ESTATE})\} \quad \square \end{aligned}$$

Double Intersection operator: the double intersection operator $\mathcal{A} \cap \mathcal{B}$ applies the intersection operator \cap of relational algebra to the maximal common subschemata of the pairs of relations R_A and R_B , belonging to \mathcal{A} and \mathcal{B} , respectively, where either $\text{Sch}(R_A) \subseteq \text{Sch}(R_B)$ or $\text{Sch}(R_B) \subseteq \text{Sch}(R_A)$.

After this first step, the standard union \cup is recursively applied between pairs of the obtained relations having the same schema, since the Double Intersection can generate several relations with the same schema.

¹ In this way we preserve selection conditions which have possibly been applied on the same tables in the definition phase of \mathcal{A} and \mathcal{B} .

Example 6. Let us now assume that the maximal area policy is applied. The view associated with the agent *role* is

$$\mathcal{R}el(\text{agent}) = \{\sigma_{IdAgent=\$ag_id} \text{AGENT}, \sigma_{IdAgent=\$ag_id} \text{AGENDA}, \\ \text{VISIT}, \text{SALES}, \text{RENT}, \text{OWNER}, \text{CUSTOMER}, \text{ESTATE}\}$$

while the view related to residential *category* is that reported in Example 3.

Consider now the context of agent *\$ag_id* specified by the chunk configuration $C = \{\{\text{agent}(\$ag_id)\}, -, -, \{\text{residential}\}\}$. The chunk obtained by applying the Double Intersection operator between $\mathcal{R}el(\text{agent})$ and $\mathcal{R}el(\text{residential})$ is the following:

$$\begin{aligned} \mathcal{R}el(C) &= \mathcal{R}el(\text{agent}(\$ag_id)) \cap \mathcal{R}el(\text{residential}) \\ &= \{\sigma_{Category=\text{Residential}} \text{ESTATE}, \text{OWNER} \times (\sigma_{Category=\text{Residential}} \text{ESTATE}) \\ &\quad \text{VISIT} \times (\sigma_{Category=\text{Residential}} \text{ESTATE}), \text{SALES} \times (\sigma_{Category=\text{Residential}} \text{ESTATE}) \\ &\quad \text{RENT} \times (\sigma_{Category=\text{Residential}} \text{ESTATE}), \\ &\quad (\sigma_{IdAgent=\$ag_id} (\text{AGENDA})) \times (\sigma_{Category=\text{Residential}} \text{ESTATE})\} \quad \square \end{aligned}$$

Note that Double Union and Double Intersection are *commutative* and *associative*.

4 View Extraction

To better illustrate our methodology, in this section we propose a real-world case study about the estate agency scenario introduced in Section 2. Suppose that, by applying the value-based mapping strategy, the designer has associated the following partial views by using the *maximal relevant area policy*:

$$\mathcal{R}el(\text{agent}(\$ag_id)) = \{\sigma_{IdAgent=\$ag_id} \text{AGENT}, \sigma_{IdAgent=\$ag_id} \text{AGENDA}, \\ \text{VISIT}, \text{SALES}, \text{RENT}, \text{OWNER}, \text{CUSTOMER}, \text{ESTATE}\}$$

$$\mathcal{R}el(\text{Residential}) = \{\sigma_{Category=\text{Residential}} \text{ESTATE}, \\ \text{OWNER} \times (\sigma_{Category=\text{Residential}} \text{ESTATE}) \\ \text{VISIT} \times (\sigma_{Category=\text{Residential}} \text{ESTATE}) \\ \text{SALES} \times (\sigma_{Category=\text{Residential}} \text{ESTATE}) \\ \text{RENT} \times (\sigma_{Category=\text{Residential}} \text{ESTATE})\} \\ \text{AGENDA} \times (\sigma_{Category=\text{Residential}} \text{ESTATE})\}$$

$$\mathcal{R}el(\text{sale}) = \{\text{SALES}, \text{OWNER}, \text{CUSTOMER}, \text{AGENT}, \text{VISIT}\}$$

$$\mathcal{R}el(\text{zone}(\$zone_id)) = \{\sigma_{Area=\$zone_id} \text{ESTATE}, \text{SALES} \times (\sigma_{Area=\$zone_id} \text{ESTATE}), \\ \text{RENT} \times (\sigma_{Area=\$zone_id} \text{ESTATE}), \text{OWNER} \times (\sigma_{Area=\$zone_id} \text{ESTATE})\}$$

Table 1. The partial views associated with the Agent *role* by using the maximal policy

```
{ create view AgentData(IdAgent,Name,Surname,Office,Address,City,Phone)
  as select * from AGENT where IdAgent=$ag_id;
create view AgentAgenda(IdAgent,Data,Hour,IdEstate,ClientName)
  as select * from AGENDA where IdAgent=$ag_id;
VISIT; SALES; RENT; OWNER; CUSTOMER; ESTATE }
```

Table 2. The partial views associated with the Residential *category* by using the maximal policy

```
{ create view ResidEstate(IdEstate,IdOwner,Category,Area,City,Province,
    RoomsNumber,BedRooms,Garage,SquareMeters,Sheet,CadastralMap)
  as select * from ESTATE where Category='Residential';
create view ResidOwner(IdOwner,Name,Surname,Type,Address,City,PhoneNumber)
  as select * from OWNER where IdOwner IN
    (select IdOwner from ESTATE where Category='Residential');
create view ResidVisit(IdEstate,IdAgent,IdCustomer,Date,VisitDuration)
  as select * from VISIT where IdEstate IN
    (select IdEstate from ESTATE where Category='Residential');
create view ResidSale(IdEstate,IdAgent,IdCustomer,Date,AgreePrice,Status)
  as select * from SALES where IdEstate IN
    (select IdEstate from ESTATE where Category='Residential');
create view ResidRent(IdEstate,IdAgent,IdCustomer,Date,RatePrice,Status,Duration)
  as select * from RENT where IdEstate IN
    (select IdEstate from ESTATE where Category='Residential');
create view ResidAgenda(IdAgent,Date,Hour,IdEstate,ClientName)
  as select * from Agenda where IdEstate IN
    (select IdEstate from ESTATE where Category='Residential') }
```

Table 3. The partial views associated with the Sale *type* by using the maximal policy

```
{OWNER; CUSTOMER; AGENT; SALES; VISIT }
```

Table 4. The partial views associated with context element Zone, by using the maximal policy

```
{ create view ZonaEstate(IdEstate,IdOwner,Category,Area,City,Province,
    RoomsNumber,BedRooms,Garage,SquareMeters,Sheet,CadastralMap)
  as select * from ESTATE where Area = $zone_id;
create view ZonaSale(IdEstate,IdAgent,IdCustomer,Date,AgreePrice,Status)
  as select * from SALES where IdEstate IN
    (select IdEstate from ESTATE where Area=$zone_id);
create view ZonaRent(IdEstate,IdAgent,IdCustomer,Date,RatePrice,Status,Duration)
  as select * from RENT where IdEstate IN
    (select IdEstate from ESTATE where Area=$zone_id
    );
create view ZonaOwner(IdOwner,Name,Surname,Type,Address,City,PhoneNumber)
  as select * from OWNER where IdOwner IN
    (select IdOwner from ESTATE where Area=$zone_id)}
```

Table 5. The views associated with the chunk configuration $\{\{agent(\$ag_id)\},-, \{residential, sale, zone(\$zone_id)\}\}$ by using the maximal policy

```
{ create view ResidZonaEstate(IdEstate,IdOwner,Category,Area,City,Province,
    RoomsNumber,BedRooms,Garage,SquareMeters,Sheet,CadastralMap)
  as select * from ESTATE where Area = $zone_id AND Category='Residential';
create view ResidZonaSale(IdEstate,IdAgent,IdCustomer,Date,AgreePrice,Status)
  as select * from SALES where IdEstate IN
    (select IdEstate from ESTATE where Area=$zone_id AND Category='Residential');
create view ResidZonaOwner(IdOwner,Name,Surname,Type,Address,City,PhoneNumber)
  as select * from OWNER where IdOwner IN
    (select IdOwner from Estate where Area=$zone_id AND Category='Residential')}
```

These partial views can be mapped to a set of CREATE VIEW constructs of SQL or database relations, as shown by the code fragments of Tables 1, 2, 3 and 4. The chunk

related to the chunk configuration $\langle \text{agent}(\$ag_id), -, -, \{\text{residential}, \text{sale}, \text{zone}(\$zone_id)\} \rangle$ can be obtained by applying the \bowtie operator to the partial views as follows:

$$\begin{aligned} \mathcal{R}el(\langle \text{agent}(\$ag_id), -, -, \{\text{residential}, \text{sale}, \text{zone}(\$zone_id)\} \rangle) = \\ \mathcal{R}el(\text{agent}(\$ag_id)) \bowtie \mathcal{R}el(\text{residential}) \bowtie \mathcal{R}el(\text{sale}) \bowtie \mathcal{R}el(\text{zone}(\$zone_id)) \end{aligned}$$

The translation into a set of SQL views is reported in Table 5.

5 Related Work

Several policies and methodologies for view definition have been suggested in the literature during the past years. More precisely, context and its dimensions have been studied in ([8,9,10,11,12,2]) from several points of views and spanning over interdisciplinary research fields ([13]). The early pioneer models and view-design proposals are for relational databases [14], but more recently, the success of XML has motivated the need of models that are independent of the underlying data model. The problem of view definition for XML has been formalized in [15,16], and some other proposals in the XML area are [17,18].

In the Semantic Web Area some view models based on ontologies have been proposed in [19].

The proposals described in [14,15,16,17,18,19] follow the classical notion of view extraction process and need to be extended to consider also user's activity, tasks, and intentions, in fact, the user's context (see [20,11,21,22]).

When focusing on the relational model of data, a few solutions have been presented to introduce the notion of context in the data being modeled, by means of additional attributes explicitly defining the context (e.g., space and time characteristics) the data belongs to ([23,24,25]); in these situations, the final goal is to make data context-aware, thus context and the data schema need be designed at the same time, foreseeing if not all values, all possible context dimensions.

In [23] the Context Relational Model is presented as an extension of the relational one: the notion of context is treated as a first-class citizen that must be modeled and can be queried; indeed, in the authors' opinion, attribute values may change under different contexts and may not exist in some other contexts. The approach is different from ours, because we study a technique to tailor data, i.e., extract and elaborate relevant portions of relational databases with respect to a notion of context, and do not propose instance changes on the basis of the context dimension values.

In [26] some conceptual operators are formalized for constructing conceptual context dependent views on XML data. In our opinion, the notion of context we use is much more articulated (mutual exclusive and orthogonal dimensions are modeled) and thus, we need to define operators both to extract views, and to combine partial views; the composition step is required to obtain views suitable to represent the relevant portion of data for a notion of context that depends on more than one dimension.

6 Conclusion and Future Work

In this paper we have presented two complementary context-driven strategies for automatic view design over a relational database schema, with the aim of selecting a portion

of the entire system with respect to one of the possible contexts for the application scenario. The methodology adopting these two approaches is based on the definition of the Context Dimension Tree, representing the contexts, and on its labeling, in order to build the views. This context-aware view extraction can be adopted in numerous scenarios, from those where small devices are considered, to those where large amount of data need be filtered to offer a more simplified and efficient access and management.

While the configuration-based scenario is already supported (based on ER and XML), we are currently building a first Java prototype environment to demonstrate the flexibility of our methodology dealing with value-based mappings. It is basically formed by two components: the first one is used to configure the labeling policy and the operators to compose partial views; the second is a graphical interface which allows the designer to assign each node its partial view.

References

1. Bolchini, C., Quintarelli, E.: Filtering mobile data by means of context: a methodology. In: Meersman, R., Tari, Z., Herrero, P. (eds.) *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*. LNCS, vol. 4278, pp. 1986–1995. Springer, Heidelberg (2006)
2. Bolchini, C., Schreiber, F.A., Tanca, L.: A methodology for very small database design. *Information Systems* 32(1), 61–82 (2007)
3. McGuinness, D.L., van Harmelen, F.: *OWL Web Ontology Language Overview*, W3C Recommendation (2004)
4. Bolchini, C., Quintarelli, E.: Filtering mobile data by means of context: a methodology. In: *Proc. 2nd Int. Workshop on Context Representation and Reasoning*, pp. 13–18 (2006)
5. Bolchini, C., Curino, C., Quintarelli, E., Schreiber, F.A., Tanca, L.: Context information for knowledge reshaping. *Int. Journal on Web Engineering and Technology* (in print 2007)
6. Bolchini, C., Curino, C.A., Orsi, G., Quintarelli, E., Schreiber, F.A., Tanca, L.: CADD: a tool for context modeling and data tailoring. In: *Proc. IEEE/ACM Int. Conf. on Mobile Data Management - Demo Session*, pp. 221–223. ACM Press, New York (2007)
7. Bolchini, C., Quintarelli, E., Rossato, R.: Relational data tailoring through view composition (submitted for publication)
8. Lenat, D.: Dimensions of context-space. Technical report, Cycorp. (1998)
9. Dey, A.K.: Understanding and using context. *Personal Ubiquitous Comput.* 5(1), 4–7 (2001)
10. Benerecetti, M., Bouquet, P., Ghidini, C.: On the dimensions of context dependence: Partiality, approximation, and perspective. In: *Proc. 3rd Int.l and Interdisciplinary Conf. on Modeling and Using Context*, pp. 59–72. Springer, Heidelberg (2001)
11. Ghidini, C., Giunchiglia, F.: Local Models Semantics, or contextual reasoning=locality+compatibility. *Artificial Intelligence* 127(2), 221–259 (2001)
12. Strang, T., Linnhoff-Popien, C.: A context modeling survey. In: *1st Int. Workshop on Advanced Context Modelling, Reasoning and Management* (2004)
13. Feng, L., Apers, P.M.G., Jonker, W.: Towards context-aware data management for ambient intelligence. In: Galindo, F., Takizawa, M., Traummüller, R. (eds.) *DEXA 2004*. LNCS, vol. 3180, pp. 422–431. Springer, Heidelberg (2004)
14. Elmasri, R., Navathe, S.: *Fundamentals of database systems*, 4th edn. Pearson/Addison Wesley (2004)
15. Abiteboul, S.: On views and xml. In: *Proc. 18th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems*, pp. 1–9 (1999)
16. Cluet, S., Veltri, P., Vodislav, D.: Views in a large scale of XML repository. In: *Proc. 27th Int. Conf. on Very Large Data Bases*, pp. 271–289 (2001)

17. Zhuge, Y., Garcia-Molina, H.: Graph structured views and their incremental maintenance. In: Proc. IEEE 14th Int. Conf. on Data Engineering, pp. 116–125. IEEE Computer Society Press, Los Alamitos (1998)
18. Liefke, H., Davidson, S.B.: View maintenance for hierarchical semistructured data. In: Kambayashi, Y., Mohania, M.K., Tjoa, A.M. (eds.) DaWaK 2000. LNCS, vol. 1874, pp. 114–125. Springer, Heidelberg (2000)
19. Volz, R., Oberle, D., Studer, R., Staab, S.: Views for light-weight web ontologies. In: Proc. ACM Symp. on Applied Computing, pp. 1168–1173. ACM Press, New York (2003)
20. Jiang, L., Topaloglou, T., Borgida, A., Mylopoulos, J.: Incorporating goal analysis in database design: A case study from biological data management. In: Proc. 14th Italian Symp. on Advanced Database Systems, pp. 14–17 (2006)
21. Rajugan, R., Chang, E., Dillon, T.S.: Ontology views: A theoretical perspective. In: Meersman, R., Tari, Z., Herrero, P. (eds.) On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops. LNCS, vol. 4278, pp. 1814–1824. Springer, Heidelberg (2006)
22. Wouters, C., Rajugan, R., Dillon, T.S., Rahayu, J.W.R.: Ontology extraction using views for semantic web. In: Web Semantics and Ontology, pp. 1–40. Idea Group Pub. (2005)
23. Roussos, Y., Stavarakas, Y., Pavlaki, V.: Towards a context-aware relational model. In: Proc. Context Representation and Reasoning - CRR'05, pp. 7.1–7.12 (2005)
24. Parent, C., Spaccapietra, S., Zimanyi, E.: Conceptual Modeling for Traditional and Spatio-Temporal Applications - The MADS Approach. Springer, Heidelberg (2006)
25. Stuckenschmidt, H.: Toward multi-viewpoint reasoning with owl ontologies. In: Proc. 3rd European Semantic Web Conference, pp. 259–272 (2006)
26. Rajugan, R., Dillon, T.S., Chang, E., Feng, L.: A Layered View Model for XML Repositories and XML Data Warehouses. In: Proc. IEEE 5th Int. Conf. on Computer and Information Technology, pp. 206–215. IEEE Computer Society Press, Los Alamitos (2005)