Towards a Context-Aware Relational Model

Yannis Roussos[⋆] Yannis Stavrakas[⋆] Vassia Pavlaki[⋆⋆]

Department of Electrical and Computer Engineering,
National Technical University of Athens (NTUA)
Athens, Greece
{iroussos,ys}@dblab.ntua.gr vpavlaki@mail.ntua.gr

Abstract. In traditional databases and information systems the number of users is more or less known and their background is to a great extent homogeneous. In distributed and heterogeneous environments however such as the Web, users do not apply the same conventions when interpreting data due to different backgrounds, knowledge or culture. Interpreting and managing data according to the context is a topic not explored in its full potential in these new environments. In this work we present the Context Relational model (CR model), a model that extends the relational model to argue also about context. The interesting part of this approach is that context is treated as first-class citizen at the level of database models and query languages. This is due to the fact that an attribute may not exist under some contexts or that the attribute may have different values under different contexts. Apart from the model description this work presents also a set of basic operations which extend relational algebra so as to take context into account.

1 Introduction

Information today is accessed and used in a global environment, where assumptions about data become less evident. Users with different backgrounds or viewpoints may interpret the same data in a different way. Moreover, the interpretation and suitability of data may depend on unpredictably changing conditions, like for example the current position of the user or the media he is using (laptop, mobile, PDA). To avoid such ambiguous situations the information provider needs to specify the context under which information becomes relevant. Conversely, information users can specify their own current context when requesting for data in order to denote the part that is relevant to their specific situation.

For instance imagine a product whose specification changes according to the country it is being exported. Or a Web page that is to be displayed on devices

[⋆] Work supported in part by DELOS Network of Excellence on Digital Libraries, IST programme of the EC FP6, no G038-507618, and by PYTHAGORAS EPEAEK II programme, EU and Greek Ministry of Education, co-funded by the European Social Fund (75%) and National Resources (25%)

[⋆⋆] Work supported by HERAKLEITOS EPEAEK programme, EU and Greek Ministry of Education, co-funded by the European Social Fund (75%) and National Resources (25%)

with different capabilities, like mobile phones, PDAs, and personal computers. Another example is a report that must be represented at various degrees of detail and in various languages. In those situations, context can be used as a viewpoint mechanism that takes into account implicit background knowledge.

In previous work [SG02,Sta03] we argued that the management of context should take place at the level of database systems in a uniform way and that consequently context should be treated as a first-class citizen in data models and query languages. In particular, relational databases (R-DBs) can no longer be seen as isolated data sources, where users are familiar with the implied assumptions necessary for interpreting the data they retrieve. Today R-DBs are widely used in Web applications for dynamically constructing Web pages that are globally available. They are also used as back-ends in systems where context information is constantly provided by sensors. It is therefore interesting to examine how the notion of context can be incorporated in a R-DBMS.

In [Sta03], it has been shown that direct incorporation of context in database management systems would have the following positive results: a) management of data according to the interpretation frame, b) ability to pose *cross-world* queries that have no counterpart in context-unaware systems, c) personalization in a flexible and uniform manner and d) direct support for managing data and schema histories.

In this work we study the problem of incorporating context in the relational database model, which has a strong formal background, and propose a novel infrastructure. We present the Context Relational model (CR model) that can be viewed as an extended relational model able to argue about context as well. The proposed model is able to accommodate information entities that manifest different *facets*, whose contents can vary in structure and value. Each facet of such a multi-faceted information entity is associated with a context, stating the conditions under which this facet holds. We then discuss some operations through example queries that demonstrate the potential of this model.

The rest of the paper is organized as follows. Section 1.1 presents related work. In Section 2 the notion of context is revised. In Section 3.1 we give a motivating example to facilitate the understanding of our model. In Section 3.2 we discuss the technical challenges of our work and the advantages of our framework. In Section 3.3 we describe in detail the CR model. Section 4 presents the basic operations of an extended algebra that incorporates context through example queries. In Section 5 we state the differences of the CR model relatively to the relational model. Finally Section 6 discusses possible future research directions.

1.1 Related Work

The notion of context was originally introduced in Frege's proposal of a principle of contextuality [Fre84]. Since then, context has been called to account for a variety of problems in different areas. Context has been used in diverse areas of computer science as a tool for reasoning with viewpoints and background beliefs, and as an abstraction mechanism for dealing with complexity, heterogeneity, and partial knowledge. A formal framework for reasoning upon a subset of a global knowledge base can be found in [GG01], which is extended in Local

Relational Model [SGMB03,BGK^{+}02] to allow for inconsistent databases and to support semantic interoperability in the absence of a global schema. Examples of how context can be used for partitioning an information base into manageable fragments of related objects can be found in [MMP95,TACS98].

In Web information systems context is often defined by giving values to a number of user-defined variables called dimensions or characteristics. By assigning values to such variables, users can constraint the information that is relevant to them by explicitly specifying the way they interpret data. In [Bro98,Yil97], Intensional HTML is introduced to address the problem of constructing and maintaining Web sites that must exist in many versions, for example depending on the language, user expertise, subscription level, screen resolution.

In multidimensional semistructured data [Sta03,SG02] similar principles are applied to represent semistructured information entities that assume different facets, with varying content (value and/or structure), under different contexts. This approach sees context as a set of constraints, which are combined through context operations to specify environments under which information obtains an unambiguous meaning. Such an environment is called a world, while constraints are expressed by assigning values to dimension variables. The work in [SGDZ04] shows how context can be used to represent and query valid time of data and histories of semistructured databases.

This perception of context has also been used in OMSwe, a Web-publishing platform described in [NP03b,NP03a]. OMSwe is based on an Object DBMS, which has been extended to support a flexible, domain-independent model for information delivery where context plays a pivotal role.

New mobile-aware applications are more effective and adaptive to user information needs without consuming too much of a users attention, by taking advantage of the dynamic environmental characteristics, such as the user's location, time, people nearby, etc. A detailed survey about exploiting context-aware computing in mobile environments is presented in [CK00]. In [DVV03], the authors propose a context-aware service discovery, describing a context-based index to support efficient retrieval of Web services whereas in [VVV^{+}03] a platform is presented for sharing context dependent data and services for mobile sources, called MobiShare.

2 Preliminaries

In our approach, a world represents an environment under which data obtain a substance. The set of parameters used to specify the world are called *dimensions*. A *context specifier* is a syntactic construct used to qualify pieces of data and specify sets of worlds (or contexts) under which these pieces hold. In this way, it is possible to have at the same time variants of the same *information entity*, each holding under a different set of worlds, or equally under a different context. The variants of an information entity are called *facets* of the entity and they may differ in value and/or structure.

Definition 1. *Let $\mathcal{D}$ be a nonempty set of dimension names and for each $d \in \mathcal{D}$, let $\mathcal{V}_d$ be the domain of d, with $\mathcal{V}_d \neq \emptyset$. A* world *$w$ with respect to $\mathcal{D}$ is a set* *of pairs (d, v), where $d \in \mathcal{D}$ and $v \in \mathcal{V}_d$, such that for every $d \in \mathcal{D}$ exactly one (d, v) belongs to w.*

In the model we present in Section 3, sets of worlds are represented by context specifiers, which can be seen as constraints on dimension values.

Example 1. Some context specifiers might be the following:

```
(a) [device=PC]
(b) [device=PDA, payment in {credit card, cash}]
```

In Example 1, context specifier (a) represents the worlds for which the dimension `device` has the value `PC`, while (b) represents the worlds for which `device` is `PDA` and `payment` is either `credit card` or `cash`. It is not necessary for a context specifier to contain values for every dimension in $\mathcal{D}$. Omitting a dimension implies that its value may range over the whole dimension domain.

The context specifier `[]` is a *universal context* and represents the set of all possible worlds, while the context specifier `[-]` is an *empty context* and represents the empty set of worlds. In [SG02,Sta03] operations on context specifiers are defined and it is presented how a context specifier can be transformed to the set of worlds it represents with respect to a set of dimensions $\mathcal{D}$. In the case where two context specifiers represent disjoint sets of worlds the context specifiers are called *mutually exclusive*.

3 A Context Relational Model

In this section we start with an example used throughout this paper and we continue with stating some of the challenges we had to face. Then we present in detail a Context Relational Model (CR model) for the representation and manipulation of information under different worlds.

3.1 Motivating Example

Example 2. Consider a Web site about digital cameras. The information provided for each camera includes the brand name, the model of the camera, a picture, the size in megapixels and the price. Customers connect to this Web site using a variety of devices ranging over desktop computer (PC), PDA and cell phone. Moreover, they can select the method of payment between Credit Card and Cash.

The data about a digital camera returned to customers depends on the browsing device and payment method. That is, a customer using a `PDA` receives a picture of lower resolution than when using a PC. When using a cell phone no picture exists and only textual information is provided. Moreover, the price of a digital camera varies according to the payment method. Evidently, digital camera is an information entity whose contents and structure in this particular example vary according to the browsing device and the method of payment.

In database terms there is a main relation `dcamera`, with attributes: `Brand`, `Model`, `MPix`, `Photo`, `Price` (see Table 1b). Customer requests are expressed as queries and the data returned correspond to the evaluation of the queries over this relation. The context of the example relation is expressed through the following dimensions: device ranging over {`PC`, `PDA`, `CELL`}, payment ranging over {`Credit Card`, `Cash`}.

The schema of relation `dcamera` and the data for a particular entity may differ between different worlds. This is due to the fact that an attribute may not exist in some worlds and that the same attribute may have different values under different worlds. Table 1a presents all the possible worlds, while Table 1b presents the worlds under which each attribute is defined.

World	**Device**	**Payment**
w_1	PC	Credit Card
w_2	PDA	Credit Card
w_3	CELL	Credit Card
w_4	PC	Cash
w_5	PDA	Cash
w_6	CELL	Cash

Attributes	**Worlds**
Brand	defined in every world
Model	defined in every world
MPix	defined in every world
Photo	defined only for worlds with browsing device in {PC,PDA}
Price	defined in every world but its values may change

Table 1. (a) On the left, all the possible worlds for Example 2 are presented and (b) on the right, the worlds under which each attribute is defined are presented

3.2 Technical challenges and comparison with related work

The relational model does not treat context and entities as first class citizens. As a result, the definition and manipulation of information that exists in multiple worlds is done at application level. This has a series of disadvantages:

1. Redundancy, as the logic and the different basic operations must be reimplemented in each application.
2. Implementations that are strictly connected to specific design decisions and the nature of each application.
3. Maintenance overhead.

There are two other approaches for dealing with context. In the first approach context is managed by the application logic at a separate level from data management leading to an inflexible and hard to maintain architecture, whereas ad-hoc solutions must be developed from scratch for every new case. In the second approach, context is managed in a mediator, at an intermediate layer between the data source(s) and the application(s). In this approach the mediator contains context metadata that are "linked" to the information parts they annotate at the sources. Queries are addressed to the mediator, which uses context metadata to navigate to the relevant data.

Handling context at the level of information sources (ie. database systems) has the following advantages:

1. Comprehensive data design: context and its relation to data are taken into account at the stage of schema design. This introduces an additional complexity to schema design, however the incorporation of context conditions

into the DB schema enables also the consistency checking of insertions and updates of context-dependent data.
2. Wider spectrum of queries: data models and query languages that directly incorporate context can be more expressive, leading to new sorts of queries (cross-world queries [Sta03]), like "show the pictures of cameras that are more expensive when paying in cash than when paying using credit card", or data management queries like "under which contexts there is no picture".
3. Efficient access: by pushing context management at the level of information sources, the database systems have complete control and can use their context-aware schema information as well as potentially especially designed access methods to improve efficiency.

3.3 Model Description

In the previous section we argued that context should be incorporated in database engines. In this next section we present in detail the proposed CR model. The basic notion of our model is the *multi-facet entity*. A *multi-facet entity* is an information entity which assumes different facets as defined under different worlds. In the remaining of the paper, when we write *entity* we mean a *multi-facet entity*. A facet $f_{i,j}$ is the variant of an entity e_i defined under a specific world w_j. A set of entities are grouped to form a *context relation*.

For a *context-relation* R, a number of attributes are defined in each world. The sets of attributes defined for R in two different worlds can (i) be the same, (ii) have a common subset or (iii) have no common attributes at all. Moreover, the same attribute A_i of a single entity can have different values in different worlds. We refer to the value of an attribute A_i in world w_j as $A_i\{w_j\}$.

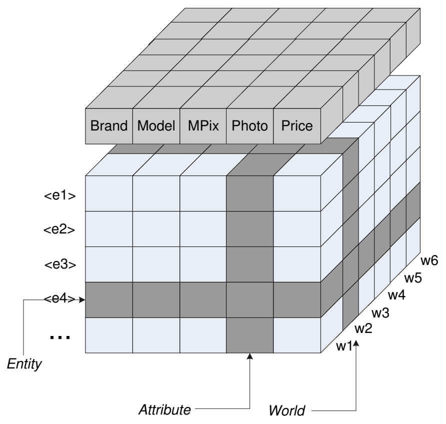

Fig. 1. The Context Relation for the digital camera example

In Figure 1 we present the *context-relation* `dcamera` of Example 2. In order to show the basic parts of our model, we highlight the entity e_4, the attribute `Photo` and the world w_2.

Entities belonging to *context-relation* `dcamera` have values for the six worlds of Table 1a. As stated in Table 1b, attribute `Photo` is not defined for worlds w_3 and w_6 (`device = CELL PHONE`), it stores a high resolution photo for worlds w_1 and w_4 (`device = PC`) and a low resolution one for worlds w_2 and w_5 (`device = PDA`). Also the value of attribute `Price` depends on the payment dimension. For example, entity e_4 has $\texttt{Price}\{w_1\} = 154$ and $\texttt{Price}\{w_4\} = 142$.

A *context-relation* can be viewed as a cube. Each entity of the relation is an horizontal slice spanning across all the worlds. Each entity slice has $|a| * |w|$ cells, where $|a|$ is the total number of attributes of the entity and $|w|$ is the total number of worlds in the system. A cell {i,j} in this slice represents the attribute $A_i\{w_j\}$. In Figure 1, we highlight entity e_4, which is a two-dimensional 5*6 horizontal slice, as *context-relation* `dcamera` has five attributes and is defined for six worlds. Attribute `Model` is the second attribute of *context-relation* `dcamera` and { `Cell Phone ,Credit Card`} is the third world, so cell {2,3} represents attribute $\texttt{Model}\{w_3\}$ for entity e4.

An attribute is a two-dimensional $|e| * |w|$ vertical slice, where $|e|$ is the total number of entities in the context dependent relation. A cell {i,j} in the slice corresponding to attribute A_k represents the value of A_k for entity e_i in world w_j. Proportionally, a world is a two-dimensional $|e| * |a|$ vertical slice. Each facet of an entity e_i is represented by the intersection of the slice entity for e_i and the world slice for the corresponding world.

Modeling entities and attributes as two-dimensional slices allows us to exhibit these relations and interpret queries over different worlds in a more meaningful way. For example, in Figure 1, we highlight attribute `Photo`. The response to a query that asks for the photos of entity e_4 is represented by the intersection of the vertical attribute slice for attribute `Photo` and the horizontal entity slice for e_4. The response to a query that asks for all the photos in world w_2 is represented by the intersection of the two vertical slices for attribute `Photo` and world w_2. Finally, the intersection of the three slices corresponds to the value of attribute `Photo` for entity e_4 in world w_2.

4 Operations

In this section we present the basic operations (*projection*, *selection*, *cartesian product*, *join* and *set operations*) of the relational algebra extended to incorporate context. To distinguish them from the classical operations, we use the term context in front of them, as for instance *context-project.* We keep the relational symbols, e.g. π for *context-project.*

Context–Project

Given an input *context-relation*, the objective is to output only the attributes specified in the condition of the *context-project* operator. The result is a new *context-relation* that has only the vertical slices that correspond to the projected attributes. The resulting context relation will still include slices for all possible worlds, however will only contain actual values for the projected part (highlighted cells in figures of this section).

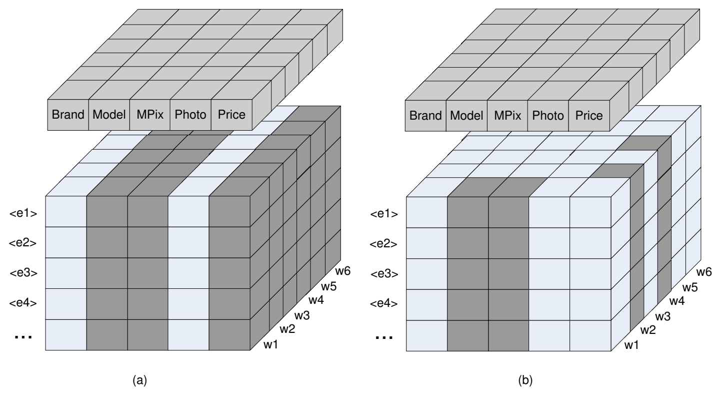

Fig. 2. a)*Context-Project* for attributes Model, MPix and Price in all worlds, b)*Context-Project* for attributes Model and MPix in world w_1 and for Price in worlds $\{w_2, w_4\}$

For example, in Figure 2(a) we see the result of projecting attributes `Model`, `MPix` and `Price`:

$$\pi_{Model[\ ],\ MPix[\ ],\ Price[\ ]} dcamera$$

Note that a context specifier follows each attribute. Specifying the context of the projected attributes is crucial when we are interested in the values of attributes only for some worlds. For each attribute, the context specifier denotes the set of worlds that will be projected.

Figure 2(b) illustrates the following query, which projects attributes `Model` and `MPix` for world w_1 and attribute `Price` for worlds w_2 and w_4:

$$\pi_{\substack{Model[Device=PC, Payment=CreditCard], MPix[Device=PC, Payment=CreditCard], \\ Price[Device=PDA, Payment\ in\ \{CreditCard, Cash\}]}} dcamera$$

World Project

This operation retains only those facets of entities that hold under specified worlds. For example, for a customer using a PC the relevant worlds are w_1 and w_4. We use the letter κ to represent the *world-project* operator:

$$\kappa_{[w_1, w_4]} dcamera$$

Entity Context–Select

The *entity context-select* operation uses context in order to express conditions that involve attribute values under different worlds. The following queries illustrate this point. Note that we use the symbol σ^{entity} to denote the *entity context-selection.* Figure 3(a) highlights digital cameras created by Kodak, with more than three megapixels.

$$\sigma^{entity}_{(BRAND[\ ]='Kodak'\ AND\ MPix[\ ]>3)} dcamera$$

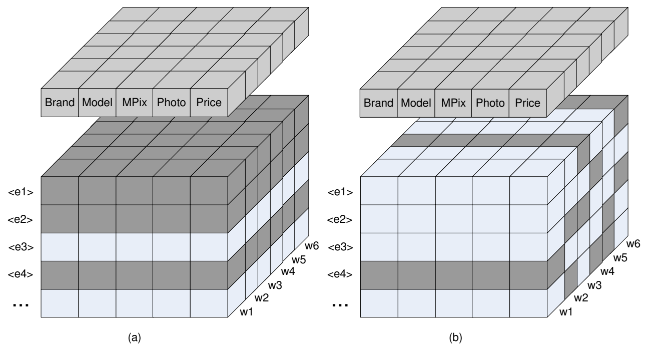

Fig. 3. (a) Entity Context–Select, (b) Facet Context–Select

A simpler case where selection criteria is restricted to the values in specific worlds is illustrated by the following example query, which selects only entities where the price is less than 250 when the customer is paying in cash:

$$\sigma^{entity}_{(BRAND[\ ]='Kodak'\ AND\ Price[Device=PC,Payment=Cash]\ <\ 250)} dcamera$$

Notice that an attribute may appear more than once in the same condition with different context specifier. The following query is a cross-world query that compares the values of the same attribute in different worlds, asking for cameras that have lower price if the customer pays in cash than using a credit card:

$$\sigma^{entity}_{(Price[Device=PC,Payment=Cash]\ <\ Price[Device=PC,Payment=CreditCard])} dcamera$$

In the previous queries each attribute in the select condition is evaluated in the set of worlds indicated by the respective context specifier. In case there is no context specifier for an attribute, we evaluate the condition to true if there exists at least one world where the condition holds. For instance, the following query requests for entities where at least one facet has `Price` less than 250 Euros:

$$\sigma^{entity}_{(BRAND[\ ]\ =\ Kodak\ AND\ Price\ <\ 250)} dcamera$$

Facet Context–Select

Users should be able to pose queries that involve selection of facets or, in other words, to select parts of an entity. This can be done using the *facet context–select* operator, denoted σ^{facet}. The σ^{facet} operator selects only the facets of an entity that satisfy the condition, instead of the whole entity as the σ^{entity} operator.

In Figure 3(b) we present the result of selecting facets with `Price` less than 500 euros. Note that the select facet operation is a basic operation in our algebra as it cannot be constructed by any combination of the other operations.

Context Cartesian Product

In a way similar to the relational model, context cartesian product creates new *context-relation*s from existing ones. For the needs of our example Figure 4(a) presents a new *context-relation* named `accessories`. The attributes defined for *accessories* are the `accessory name`, the `model` of the corresponding digital camera that the accessory fits in, a `description` of the accessory and its `price`. Attribute `model` serves as a foreign key to the `dcamera` *context-relation.*

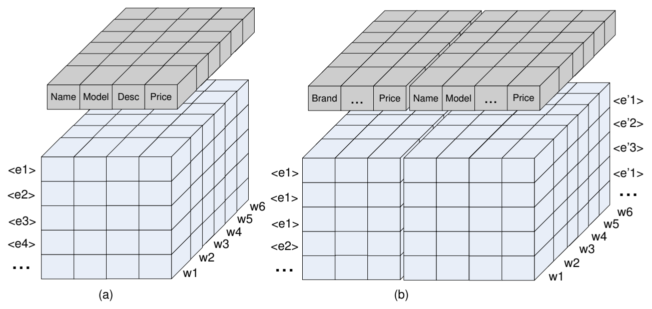

Fig. 4. a) context-relation *accessories*, b) Cartesian project

The cartesian product of *context-relations* `dcamera` and `accessories` is presented in Figure 4(b). The attributes of the resulting *context-relation* are the attributes of `dcamera` and `accessories`. Entities of the new relation are constructed by combining each entity of `dcamera` with all the entities of `accessories`.

Context-Join

A *context-join* is defined as the composition of the *context cartesian product* and the *entity context-select* operations, as in the relational model. As an example, consider a query that asks for Kodak digital cameras with cheap 200mm lens (e.g. where $accessories.Price < 0.2 * dcamera.Price$). Apart from the *context-join*, which adds no new complexities in our algebra, there is a need for a more restrictive join operation, named *context natural join.* This operation is necessary for selecting entities where (dcamera.Model = accessories.Model) for all worlds.

Set operations

Set operations like *union*, *intersection*, *difference* and *division* are defined only for *context-relations* that are *union compatible.* In order for two *context relations* to be *union compatible*, they must have exactly the same attributes defined under each world. The semantics is the same as in set theory, but with entities as the basic set elements.

Illustrating example

The following example brings together many of the operations presented in this section. Consider a customer who uses a PC and wants to buy a Kodak camera costing less than 400 Euros. He/she also wants to buy accessories for this camera, so for all selected cameras he/she then requests to see all available accessories. Finally, he/she asks the price using cash for camera and credit card for accessories. In database terms he/she requests `dcamera.Price` for cash and `accessories.Price` for credit card:

$\texttt{Ctx-Rel1} \leftarrow \textit{context cartesian product}(dcamera, accessories)$

$\texttt{Ctx-Rel2} \leftarrow \sigma^{entity}_{(dc.Model[\,] = ac.Model[\,]\ AND\ BRAND[\,]='Kodak'\ AND\ dc.Price[Device=PC, Payment=Cash]<400)} \texttt{Ctx-Rel1}$

$\texttt{Result} \leftarrow \pi_{dcamera.Model[Device=PC,Payment=Cash],\ dcamera.Price[Device=PC,Payment=Cash],\ accessories.Name[Device=PC,Payment=CreditCard],\ accessories.Price[Device=PC, Payment=CreditCard]} \texttt{Ctx-Rel2}$

5 CR vs Relational model

In this section we discuss the differences of the CR model relatively to the relational model. Although relational model can be in principle used to represent context dependent information, context as such cannot be handled in database level but through the use of application logic. In the CR model context is treated as a first class citizen added at the level of databases, enabling a uniform approach to context and the definition of context aware structures and operations.

In the CR model, a world slice w_i is a classical relation as in the relational model. The tuples of this relation would correspond to the facets of the entities for this world. One can argue that instead of a *context-relation* that is defined in k worlds we could have a set of k relations (as defined in the relational model). Another approach would be to create a single denormalized relation with as many attributes as the sum of the number of attributes defined in each world.

The main inconvenience of the relational model is that there is no way to define the relation between the same attribute under different worlds. Regardless of whether a *context-relation* is defined by k different relations or a single denormalized relation, the fact that two attributes represent the same attribute of an information entity under different worlds cannot be modelled.

If we decompose a *context-relation* to a series of relations, the link between facets that consist a single information entity is lost. This link is used in the CR model to formulate cross-world queries. Similarly, in the case of a single denormalized relation there is no way to specify the attributes that belong to a specific world and ask intra-world or cross-world queries. Moreover, maintaining a large number of relations (normalized approach) or relations with hundreds of attributes (denormalized approach) adds a great deal of complexity in the process of designing or altering the schema representing a context aware database.

6 Conclusions and future work

In this work we studied the incorporation of context into relational database systems and we presented the CR model that can argue about context as well. Our primarily aim was to allow the management of context to take place at the level of database systems. The proposed framework should also model heterogenous environments where users may receive different responses for the same query depending on the context. This is a consequence of our view of the model, where an attribute may not exist in some worlds or the same attribute may have different values under different worlds. The challenging part of this work consisted on how to extend the classical operations of relational algebra to include context and enable users to pose cross world queries. As future work we plan to:
- Extend the relational calculus and algebra to incorporate context.

- Propose a context-aware query language for the CR model.
- Demonstrate possible uses of our approach through example applications. In previous work, context has been used to represent time [SGDZ04]. In this case it could be possible to use the CR model for representing and querying histories of data. Moreover, in [GSK+01] the authors have shown how context can be used to achieve personalization in a uniform way at the database level.
- Design efficient access methods to take context into account.

References

[BGK+02] P. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. Data management for peer-to-peer computing: A vision. In *WebDB*, 2002.

[Bro98] G. Brown. Ihtml 2: Design and implementation. In *Proceedings of the 11th International Symposium on Languages for Intensional Programming*, 1998.

[CK00] G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dartmouth College Computer Science, 2000.

[DVV03] C. Doulkeridis, E. Valavanis, and M. Vazirgiannis. Towards a context-aware service directory. In *Proceedings of the 4th VLDB Workshop on Technologies for E-services (TES'03)*, 2003.

[Fre84] G. Frege. Die grundlagen der arithmetik. *Koebner, Breslau*, 1884.

[GG01] Ch. Ghidini and F. Giunchiglia. Local model semantics, or contextual reasoning = locality + compatibility. *Artificial Intelligence*, 127:221–259, 2001.

[GSK+01] M. Gergatsoulis, Y. Stavrakas, D. Karteris, A. Mouzaki, and D. Sterpis. A web-based system for handling multidimensional information through mxml. In *ADBIS*, 2001.

[MMP95] J. Mylopoulos and R. Motschnig-Pitrik. Partitioning information bases with contexts. In *CoopIS*, 1995.

[NP03a] M. C. Norrie and A. Palinginis. Empowering databases for context-dependent information delivery. In *CAiSE'03 Workshop on Ubiquitous Mobile Information and Col-laboration Systems (UMICS'03)*, 2003.

[NP03b] M. C. Norrie and A. Palinginis. From state to structure: an xml web publishing frame-work. In *CAiSE'03*, 2003.

[SG02] Y. Stavrakas and M. Gergatsoulis. Multidimensional semistructured data: representing context-dependent information on the web. In *CAiSE02*, 2002.

[SGDZ04] Y. Stavrakas, M. Gergatsoulis, Ch. Doulkeridis, and V. Zafeiris. Representing and querying histories of semistructured databases using multidimensional oem. *Information Systems*, 29:461–482, 2004.

[SGMB03] L. Serafini, F. Giunchiglia, F. Mylopoulos, and P. Bernstein. Local relational model: A logical formalization of database coordination. In *CONTEXT*, 2003.

[Sta03] Y. Stavrakas. *Multidimensional semistructured data: representing and querying context-dependent multifacet information on the web*. PhD thesis, National Technical University of Athens, 2003.

[TACS98] M. Theodorakis, A. Analyti, P. Constantopoulos, and N. Spyratos. Context in information bases. In *CoopIS*, 1998.

[VVV+03] E. Valavanis, C. Ververidis, M. Vazirgiannis, G. Polyzos, and K. Norvag. Mobishare: Sharing context-dependent data and services from mobile sources. In *Proceedings of the IEEE/WIC International Conference on Web Intelligence*, 2003.

[Yil97] T. Yildirim. Intensional html. Master's thesis, Department of Computer Science, Univerity of Victoria, 1997.