Bridging Relational Databases to Context-Aware Services

Elisabeth Kapsammer[2], Wieland Schwinger[1], Werner Retschitzegger[2]

[1] Department of Telecooperation (TK), Johannes Kepler University Linz
Altenbergerstr. 69, A-4040 Linz, Austria
wieland.schwinger@jku.ac.at

[2] Department of Information Systems (IFS), Johannes Kepler University Linz
Altenbergerstr. 69, A-4040 Linz, Austria
ek@ifs.uni-linz.ac.at
werner@ifs.uni-linz.ac.at

Abstract. The provision of context-aware services in terms of the anytime/anywhere/anymedia paradigm requires the consideration of certain context properties in terms of time, location, device, or simply the user's preferences. The specific characteristics of context, comprising among others large amounts of data, frequent and concurrent updates and heterogeneous sources often in terms of XML schemata, demand for an efficient management of context data. Relational databases seem to be a promising key technology for the efficient storage, retrieval, and processing of XML-based context data at the server-side. This is not least since relational databases allow to seamlessly combine context data with existing legacy data, hereby constituting a major advantage compared with native XML databases or object-oriented databases. The focus of this paper is twofold. First, requirements for managing context data are elaborated from a database perspective. Second, based on this, a generic approach called X-Ray, mediating between existing XML-based context and content stored in relational databases, is presented. With this approach, maintainability and changeability of context data is enhanced, since the mapping knowledge is not hard-coded but rather reified within a meta schema. The meta schema allows to automatically compose context data out of the relational database when requested and decompose them when they have to be stored.

1 Introduction

Currently, we are facing a new generation of web applications being characterised by the *anytime/anywhere/anymedia paradigm*, thus providing *ubiquitous access* to services, turning e-commerce into *m-commerce*. New opportunities and challenges for web applications are offered in terms of *time-aware* [24], *location-aware* [17], *device-aware* [1], and *network-aware* [2] services which can be *personalized* for a certain user or user group [25]. The pre-requisite for realising such services is awareness of their *context*[1]. One must understand what context is to determine its relevancy and how it can be exploited for adaptation purposes. Knowledge about the device used,

[1] The notion of *context* can be found in various fields of computer science, for an overview, cf., e.g., [4].

e.g., its display resolution, would allow to render the graphical representation of a service accordingly, whereas information about location and time of access together with user preferences would allow to provide more accurate services taking into account the current situation of use.

The *kind of context* determines its specific characteristics. Some kinds of context are characterized by large amounts of data and frequent, potentially concurrent updates (e.g., the current location context), while others are relatively stable over time (e.g., a certain geographical topology or user preferences). Typically, context data stems from different sources, such as sensors or GIS (Geographical Information Systems) and is delivered in heterogeneous formats, most often using different kinds of XML schemata (e.g., CC/PP for device context [41] or GML for location context [30]). These characteristics of context heavily demand for an efficient management approach with respect to storage, retrieval, and update of context data. Since they constitute the basis for deciding on the technology for managing the context data, they will be discussed in detail in Section 2. Because of these characteristics we mainly focus on a server-side management of context data.

The employment of database systems (DBS) would allow to handle both, efficient and concurrent retrieval as well as update of large amounts of context data in a consistent way on a large distributed scale. Regarding the kind of data model used by the DBS as basis for storing XML-based context data, one can distinguish three basic alternatives [12]. First, *special-purpose or native DBS* are particularly tailored to store, retrieve, and update XML data by using XML itself as underlying data model. Examples thereof are research prototypes such as Lore [15] as well as commercial systems such as Tamino [31]. Second, because of their rich object-oriented data modelling capabilities, *object-oriented DBS* such as Versant [37] as well as *object-relational DBS* such as Oracle [18] are well-suited for storing XML data. Despite of those advantages, we adhere to the third alternative for managing context data, namely *relational DBS*, for the following reasons.

Currently, a significant amount of data is stored in pre-existing relational database systems (RDBS) and will continue to be used by existing applications in the future [14]. At the same time, there is an increasing demand to seamlessly combine such existing relational content with dedicated context data, thus enhancing the adaptation possibilities of context-aware services. Data about customer relationship management (CRM), for example, can be used as basis for user profiles, address data of subsidiaries can flow into location profiles. Besides this possibility to make legacy data already stored within RDBS easily available for context processing, the usage of RDBS for storing context data would provide several other advantages such as reusing a mature technology, seamlessly querying data represented in XML and relations and deducing higher-level context information based on pre-defined functionality like the spatial component of SQL/MM [34].

On the other side context data is often represented and exchanged using heterogeneous XML schemata. Therefore, it is a major demand to mediate between various RDBS schemata at the one side and various XML schemata at the other side. For these reasons, this paper focuses on a generic approach for mediating between context data stored within RDBS and context data represented using XML. Based on this focus, the structure of the paper is as follows: Section 2 gives an overview about the specific characteristics of context and reasons about the benefits of using RDBS when dealing with these peculiarities. Based on this background, Section 3 proposes a generic mediation mechanism called X-Ray, which allows to flexibly map XML-based context data into a RDBS schema. Section 4 discusses related work and finally, Section 5 concludes the paper with a short summary.

2 Requirements for Managing Context Data

The notion of context can be found in various different fields of computer science such as user interfaces being either adaptive [16] or even intelligent and advisory [6], information filtering and recommender systems [26], adaptive hypertext and hypermedia [5], and mobile computing [1] (for an overview, cf., e.g., [4]). In this paper, context subsumes all information that can be used to characterize the situation of use from which adaptation of services can be inferred. Note that this understanding of context is broader than traditionally regarded in context-aware systems focusing mainly on sensory context information [17] or personalization on basis of user preferences [25].

Context characteristics which are relevant for the management of context data can be categorized into *scope* of context, its *representation* and *acquisition,* as well as the *access mechanism* used (cf. also [13], [25], [32]). In general, concerning scope and representation, RDBS are mainly advantageous because of their main focus on handling large amount of data, whereas regarding acquisition and access, RDBS are best suited since they provide various comfortable and optimised declarative and procedural access mechanisms. In the following, the specific benefits of using RDBS for the management of context data are explicitly discussed wherever applicable.

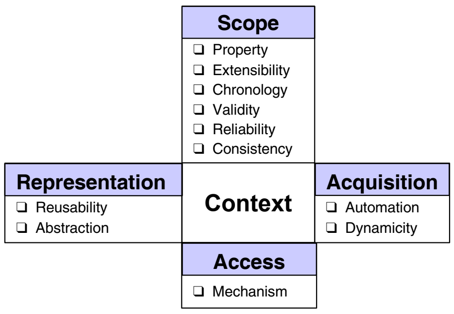


Fig. 1. Characteristics of Context

2.1 Scope of Context

The scope of context comprises not only the different context *properties,* supported by the system, together with the ability to *extend* them, but also the time dimension of context in terms of *chronology* and *validity,* and certain constraints concerning its *reliability* and *consistency*.

Property. Although the relevant context is dependent on the context-aware service at hand, a set of built-in context properties should be supported. It has to be emphasized that the characteristics discussed further on are applicable to each of these context properties. The following context properties are most often found in literature:

Location: Location copes with the need for mobile computing and location-aware services by capturing information about the location from which a service is accessed.

Time: This context property allows to adapt services with respect to certain timing constraints, e.g., opening hours of shops or timetables of public transportation.

Device: This context property refers to the demand for multi-channel delivery and provides basic information about the hardware and software capabilities of the device accessing the service.

Network: To allow adaptation on basis of the network it is necessary to provide network context in terms of, e.g., bandwidth or package losses.

User: Information about the user, e.g., demographic data, knowledge, skills and capabilities, interests and preferences, goals, and plans takes into account the necessity of personalization. In this respect, also the issue of privacy comes into play, requiring the confidential handling of personal data.

Using RDBS for the management of context is especially beneficial for location, time, and user properties. Concerning location, the SQL2003-Standard SQL/MM [34] offers predefined spatial data structures for geometry, location, and topology and appropriate methods. These methods allow, e.g., to convert between geometries and external data formats, to retrieve properties or measures from a geometry, or to compare two geometries with respect to their spatial relation [34]. Concerning the management of time context in RDBS, several proposals exist in order to realise the concepts of *validity time*, *transaction time,* and a combination thereof in terms of *bi-temporal mechanisms* [35] (cf. context validity and chronology below). Finally, concerning the user context, sophisticated *authorisation mechanisms* offered by RDBS are especially useful for realising privacy concerns.

Extensibility. Certain services may require also additional context properties which are not built-in (e.g., current outside temperature or heartbeat rate). It is not possible to foresee what kind of context properties that might be, since the list of context properties is virtually unlimited. Thus, it is required that built-in context properties can be easily extended by additional ones. RDBS allow to easily incorporate additional context properties using appropriate *schema modification operations* while preserving the consistency of existing ones.

Chronology. Practice has shown that it is useful to broaden the view on context by considering not only the *current context* at a given point in time, but also *historical information*. This is necessary to be able to identify changes in context over time. For example, since the bandwidth might be constantly changing, it is more important to be able to trace the average bandwidth instead of just having information about the latest one. In contrast, allowing to adapt towards a restricted display size requires information of the *current* device, only. Besides historical context, it might be useful to anticipate possible *future states* of a context, too. For example, concerning video streaming, it is not only relevant how the bandwidth changed in the past, but also how the bandwidth will develop or how stable it can be considered in the future, to be able to tune the resolution of the video accordingly, thus guaranteeing a constant video stream. This implies that context may accumulate to large amounts of data over time, which can be efficiently stored by RDBS due to the support of various physical storage structures such as clusters and indexes [33]. Since context can be regarded as a vector of context properties over time, the temporal database concept of *transaction time* seems to be - as already mentioned – best suited for an efficient representation thereof [35].

Validity. Context properties may not be valid during the complete period until they are updated on basis of the environment. In a mobile scenario an example would be, that, if location information does not change within a certain period, the device might not be online any longer and thus, the location information might no longer be valid. Another example would be the validity of opening hours, which, for example, may change during the seasons. Thus, it may be required to specify the validity period during which a context is valid. Again, the temporal database concept of *validity time* seems to be applicable in this respect [35].

Reliability. The characteristic of reliability is closely associated with validity. In many cases, it is not possible to be absolutely certain about the validity of a certain context property. For example, the precision of the location position is dependent on the number of satellites which are available for measuring the position. Because of this, uncertainty needs to be explicitly regarded, which can be done, e.g., by *uncertainty mechanisms* offered by database systems [36].

Consistency. Context properties must conform to certain consistency constraints which are, of course, heavily dependent on the kind of context property considered. These can be *static consistency constraints*, preventing e.g., that temperature measurements are out of a certain range or *dynamic consistency constraints*, which can, e.g., assure that position changes need to be within a certain movement vector over time, which represents the maximum walking speed of persons. RDBS are especially suited for representing such constraints by a variety of mechanisms, comprising the possibility of specifying *declarative constraints* (e.g., primary/foreign keys or check clauses), the definition of *procedural constraints* in form of *triggers,* and the specification of semantic processing units in form of *transactions* [38].

2.2 Representation of Context

The representation of context comprises two important issues, namely mechanisms for enhancing the *reusability* of the context representation and the level of *abstraction* at which context is represented.

Reusability. A key requirement concerning representation is the notion of reusability of existing context. Context-aware services may draw from various context sources like, e.g., GIS (Geographic Information Systems) from third-party providers or CC/PP-representations [41] from device vendors. Additionally, existing legacy content may also serve as context. For example, user profiles can be enriched by existing customer relationship management data. It has to be emphasized, that in such cases the border between *context* and *content* data gets blurred. Such existing context and content sources can be integrated into a common context database by using *distributed database technology* [29].

Abstraction. According to the level of abstraction, at which context properties are represented, it has to be distinguished between *physical context* and *logical context*.[2] Whereas physical context is at a very low level of abstraction and can be directly sensed from the environment (e.g., location in terms of a mobile phone's cellID), logical context enables to enrich its semantics thus making it meaningful for context-aware services (e.g., a street name). Logical context can be provided in terms of profiles (e.g., describing certain hardware characteristics), which is most often the case, but can also be built from existing physical or other logical context by applying different kinds of abstraction or inference mechanisms, (e.g., several geographical positions can be grouped representing a town) [11]. Inference of logical context can be done centrally within RDBS using declarative means in terms of arbitrary complex *views* or non-declarative mechanisms using *stored procedures*. For the sake of efficiency, views containing inferred logical context can be materialised, too [29]. Concerning procedural inference, RDBS provide a set of pre-defined methods, e.g., for carrying out spatial inference (cf. above).

2.3 Acquisition of Context

The acquisition of context can be characterised by the degree of *automation,* considering *who* is responsible for acquiring the context, and the degree of *dynamicity* in terms of *when* the context is acquired.

Automation. Concerning the acquisition of context, first it has to be defined *who* is in charge for gathering appropriate context, be it either a human *(manual acquisition),* the system *(automatic acquisition),* or a combination thereof *(semi-automatic acquisition).* For context-aware services, it is desirable to gather as much context data as possible automatically, to reduce user interaction. Physical context properties may be sensed directly from the environment, logical context may be automatically computed on the basis of other context data available. Bandwidth available in the future is an example of context constructed automatically on bases of the history of bandwidth. Semi-automatic means that automatic acquisition of a physical or logical context is accompanied with information entered manually by, e.g., a user, a designer, or the vendor of a device. Employing RDBS, acquisition of certain logical context data can be done by using *database triggers,* which infer logical context data from physical and other logical context data, automatically.

Dynamicity. Another important aspect is *when* context acquisition takes place. Considering the frequency of context changes, context can be either *static*, i.e., determined once at application start up (e.g., the device used to select the appropriate interaction style), without considering any further changes or *dynamic*, i.e., determined on every change during runtime (e.g., the bandwidth to adapt the resolution of an image on the fly). Especially in the case of dynamic context acquisition, RDBS are advantageous since context updates, which might be done concurrently from several different sources, can be consistently processed by using transaction mechanisms. Furthermore, RDBS can be optimised towards high update frequency of, e.g., sensor data [33].

[2] Note, that we use the term context for depicting both, physical and logical context, not least since all characteristics discussed are applicable to both, physical and logical context.

2.4 Access to Context

Mechanism. Context has not only to be acquired and represented in a proper way, but there must also be appropriate mechanisms in order to make context accessible to the services using the context. In this respect one can distinguish between *pull-based* and *push-based approaches*. Whereas the former require to pull context data as soon as it is required, the latter provide the current context data (to the interested "clients" in case of a subscription mechanism) as soon as a context change occurred. It has to be emphasised that, for flexibility reasons, a combination of both would be most desirable. RDBS can support the efficient retrieval of context data by means of various *optimisation techniques* [33]. In addition, *database triggers* can be employed for realising push-based access to context data.

3 Architecture for Bridging RDBS to Context-Aware Services

This section provides an overview of our system for flexibly bridging RDBS to context-aware services. Since we regard XML to be the most common form for exchanging and processing context data, we first discuss different alternatives for storing XML in RDBS. After that, the basic architecture of our system is sketched out, differentiating between initialisation time components and runtime components..

3.1 Alternatives for Storing XML in RDBS

There exist three basic alternatives for storing XML data in an RDBS. The most straightforward approach would be to *store XML data as a whole* within a *single database attribute* using a simple CLOB attribute (cf., e.g., [7]) or a dedicated XML data type (cf., SQL/XML-standard [9]). Another possibility would be to *decompose XML documents* in some way (“shredding”), e.g., into a *graph structure* and store them into appropriate database tables (cf., e.g., [12]). The third approach is that the *structure of the XML data* in terms of, e.g., a DTD *is mapped to a corresponding relational schema* wherein XML documents are stored according to the mapping, cf., e.g., [10]. Only the last of these alternatives allows to really exploit the features of RDBS such as querying mechanisms, optimisation, transactions, and the like, as pointed out in the previous section to be needed for managing context data. Thus, this approach is further used in this paper.

3.2 Employing X-Ray for Mediation

For mediating between context data represented in terms of XML and context data stored in RDBS, we use an approach called X-Ray (Integrating XML and Relational Database Systems) [23]. In fact, X-Ray constitutes a middleware for a generic and thus flexible mapping of XML schemata to relational schemata.

Meta Schema. The core component of X-Ray is a *meta schema* which is used to store information about the schemata to be mapped including the necessary user-defined mapping knowledge. In this way, it constitutes the key mechanism for the system's genericity. The main task of this meta schema is to mediate between heterogeneous concepts at the XML side and the relational side. Thus, it provides the basis for automatically composing XML data out of the relational database when requested and decomposing them when they have to be stored. Basically, the meta schema consists of three parts describing the relevant meta knowledge (cf. Figure 2).

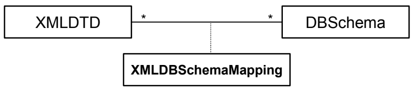


Fig. 2. Components of the X-Ray Meta Schema

The *DBSchema* part is responsible for storing information about the relational schemata used for storing context data or other legacy data relevant for the context. That means, it contains information about relations, database attributes, relationships, and joins used in those schemata. Analogously, the *XMLDTD* part stores schema information about the XML-based context data as specified by means of DTDs (Document Type Definitions), i.e., element types, XML attributes, and the composition structure[3]. Finally, the *XMLDBSchemaMapping* part stores the knowledge about mappings between DTDs and relational schemata, whereby a single DTD may be mapped to several relational schemata and vice versa. For a more detailed discussion of the meta schema it is referred to [20].

Initialisation Phase - Mapping Knowledge Editor. Before X-Ray can be used for storing and retrieving XML data, the mapping knowledge has to be specified by means of a *mapping knowledge editor* (cf. *Mapping Knowledge Layer* in Figure 3).

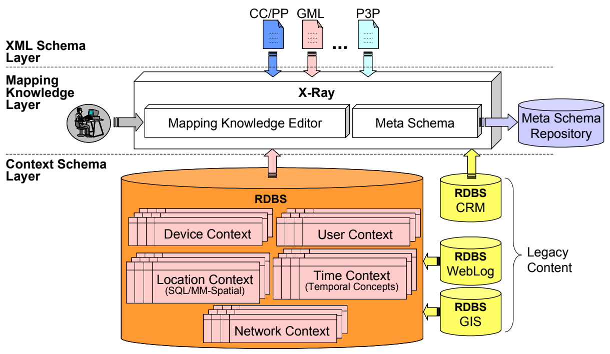


Fig. 3. Architecture of X-Ray for the Initialisation Phase

On basis of the RDBS schema and the DTD of the context data, the user may interactively specify the required mappings. It has to be emphasized, that the generation of mapping knowledge without user interaction is problematic in case of schemata developed independently of each other. Such an automatic generation would be especially feasible for the simple case where one schema should be derived from another, already existing schema, which is, however, not the focus of X-Ray. Nevertheless, approaches to automatic schema generation could be employed to support the map-

[3] Note, that although the meta schema of X-Ray, is currently extended towards capturing also the XML Schema standard [43], the X-Ray prototype supports DTDs, only [42].

ping process [3]. After specifying the mapping, the mapping knowledge is stored within the meta schema repository. This repository contains the meta schema which is in fact realised as a set of relational database tables, thus allowing for storage and retrieval of the mapping knowledge itself.

As can be seen in Figure 3, employing X-Ray for managing context data allows, on the one side, to use arbitrary XML-based context representations, e.g., CC/PP [41] for device context, GML [30] for location context, or P3P [45] for user context (*XML Schema Layer*). On the other side, the representation of context data in the database can use relational schemata, which are especially designed for efficient storage purposes and which are based on dedicated RDBS concepts such as SQL/MM-spatial for location context and temporal database concepts for time context (*Context Schema Layer*). For an exemplary relational context schema it is referred to [21]. In Figure 3 it is exemplified, that the RDBS contains schemata for the different context properties, as described in Section 2. Finally, legacy content stored in relational databases can be incorporated directly via X-Ray (cf. CRM) or indirectly using distributed database technology [29] (cf. WebLog and GIS databases in Figure 3).

Runtime Phase – Composer/Decomposer. The composer/decomposer serves for storing and retrieving context data and therefore performs all necessary transformations based on the mapping knowledge stored in the meta schema (cf. *Mapping Layer* in Fig. 4).

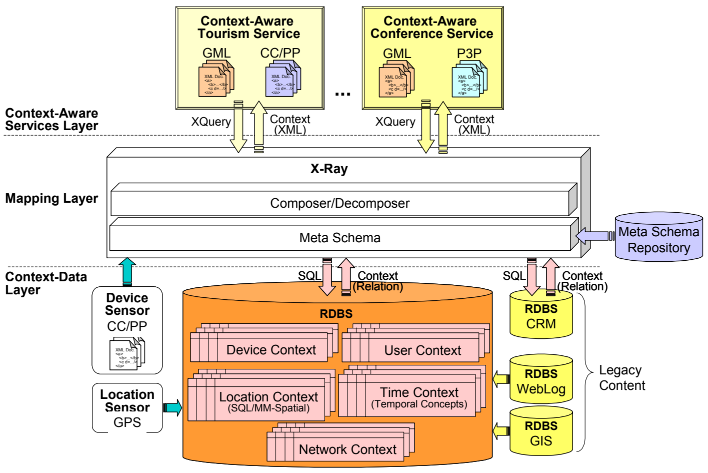
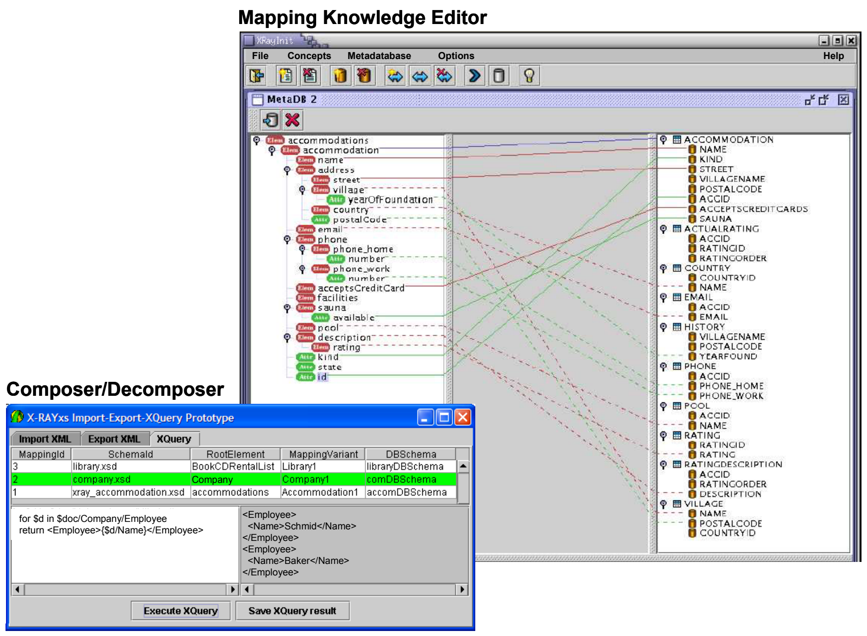


Fig. 4. Architecture of X-Ray for the Runtime Phase

During runtime, as an initial step, the relational mapping knowledge is retrieved from the meta schema repository and transferred into main memory in form of an object-oriented representation (in fact, Java data structures), herewith allowing an efficient context data transformation. After that, XQueries [44] can be issued in order to retrieve (partial) context data stored within the RDBS from a context-aware tourism service or a context-aware conference service, just to mention two examples commonly found in literature (cf. *Context-Aware Services Layer* in Figure 4). Those context-aware services may require the context information in form of, e.g., GML-, CC/PP- or P3P-formats. Utilizing the mapping knowledge, the query is decomposed into corresponding SQL queries on the relational database (cf. *Context-Data Layer*). In turn, the response from the RDBS is used by X-Ray to compose the XML-formatted context data out of relational data.

Finally, context updates can be done either directly via the RDBS, as exemplified in Fig. 4 by a location sensor (cf. *Context-Data Layer*) or, in case that context data is available in XML, by further exploiting the benefits of X-Ray via the X-Ray decomposer. In Fig. 4, as example, the device sensor provides context data in form of CC/PP, based on a vocabulary like UAProf [40], which is decomposed and stored in the device context relation of an RDBS via X-Ray.

Prototype Implementation. A first prototype of X-Ray, implemented on basis of Java and Oracle9i is already operational. Fig. 5 shows the mapping knowledge editor used during the initialization phase and the interface of the composer/decomposer responsible for storing and retrieving context data in the course of the runtime phase.



MappingId	SchemaId	RootElement	MappingVariant	DBSchema
3	library.xsd	BookCDRentalList	Library1	libraryDBSchema
2	company.xsd	Company	Company1	comDBSchema
1	xray_accommodation.xsd	accommodations	Accommodation1	accomDBSchema

```
for $d in $doc/Company/Employee
return <Employee>{$d/Name}</Employee>
```

```
<Employee>
  <Name>Schmid</Name>
</Employee>
<Employee>
  <Name>Baker</Name>
</Employee>
```

Fig. 5. Mapping Knowledge Editor and Composer/Decomposer of X-Ray

The mapping knowledge editor displays on the left hand side a certain schema for XML documents by means of graphically representing a DTD, whereas the right hand side shows a relational schema comprising relations together with their attributes. Mappings between these schemata are depicted by lines connecting them in an appropriate way. The composer/decomposer shows the choice of a certain mapping variant between two schemata and the selection of a particular root element. With respect to this choice, an XQuery can be issued and the corresponding result is displayed. For a more detailed description of the architecture of X-Ray it is referred to [22].

4 Related Work

Several approaches for providing context data have been already proposed, each of them having different origins and pursuing different goals when dealing with the unique characteristics of context data (for an overview, cf., [22]). This section describes approaches most relevant with respect to context data management, stemming from research but also from commercial vendors.

Context Toolkit [8]. The context toolkit's architecture encapsulates context management such that the context-aware services do not need to worry about how the context data is sensed, predicted, or aggregated. Context-aware services can access context from so called “widgets” using traditional poll and subscribe mechanisms. The context widgets automatically store all the context they sense and make this history of context data available to any interested services. By default, the Context Toolkit uses HTTP as network protocol and XML for representing context data, thus achieving a certain level of interoperability. The context properties supported comprise location, time, and user. There is an explicit separation between physical and logical context. Physical context is represented by so-called context widgets, which hide the details of actual context acquisition (e.g., done by means of sensors) and provide a uniform interface to components using context, thus allowing extensibility. In contrast to our approach, the Context Toolkit focuses mainly on sensing and aggregating physical context data. Existing legacy content as well as efficient, persistent storage of content data is not focused on.

IBM Transcoding Publisher [19]. IBM WebShere [19], is a commercial web application development platform. The Transcoding Publisher is part of IBM WebShere, supporting multi-channel delivery by adapting services towards different client capabilities, and bases on the "InfoPyramide" described in [27]. Additionally, IBM WebShere Personalisation and WebSphere Everyplace Suite offer a personalisation component and a component enabling to realise location-based services, respectively. The context comprises location, device, network, and user. The physical context information is dynamically enriched through separated components, called “Request Editors”, thus incorporating logical context at request. The logical context information is provided through profiles which are maintained by the developer through dedicated tools. In this way Request Editors represent logical context information in a reusable way. Explicitly, the current context of a request is considered only. Monitoring components are foreseen in the platform framework which would allow the developer to consider also historical information as well as the automatic generation of logical context information but no explicit support is given. No validity constraints are addressed in the approach.

Norrie et al. Norrie et al. propose a context engine which can be used to adapt content stored within an object-oriented content management system. In the reverse direction, the approach also utilizes content data stored within the content management system as part of the context information, an idea also followed by X-Ray. They support a pull- and push-based context acquisition mechanism which abstracts from the concrete sensing of context data. Instead of providing a certain context schema, a Prolog-based context definition language is suggested which can be used for defining arbitrary context. One particular focus lays on this specification of context in form of reusable context types within the content management system. The focus of the X-Ray approach presented in this paper differs in that it is not primarily aiming at defining new context types. Rather it provides an infrastructure for mediating between multiple existing context schemata as well as between context schemata and content schemata. A further difference appears to be that it is a dedicated goal of X-Ray to manage context data based on existing infrastructures and standards, such as XML, XQuery, RDBS, and SQL-standards like SQL/MM-spatial.

Oracle Wireless Edition [39]. Oracle offers a product called "Oracle Application Server Wireless" which is part of the Oracle Application Server. Originating in the area of multi-channel delivery, the focus is to provide a platform which enables not only to develop new web applications independent of any device, but also to adapt existing web applications to be used from various devices using a proxy-based architecture. Oracle supports location, device, and user context at a physical and a logical level in terms of profiles, stored within relational tables, which is similar to X-Ray. Physical context properties are acquired automatically or manually at runtime. Location context can be manually defined by the user (in case that automatic localisation is not possible) using so-called location marks which associate logical location information, i.e., an address with coordinates. Device context and user context is identified automatically by examining the request header. In case that the user context is not available, an explicit login procedure is used. Logical user context is supported by allowing to specify very basic user-related data and application-related data (e.g., activated services), logical device context is restricted to some very basic information about devices (e.g., screen-width and –height). It is neither possible to incorporate additional physical or logical context properties into the system nor to change the existing schema for logical context information, with the exception of user context supporting an appropriate API. Chronology of context or a validity period are not supported, context is accessed in a pull-based manner.

X-Ray-based Context Mediation. In contrast to these approaches, employing X-Ray for mediating between XML-based context data and relational schemata offers several benefits. First, X-Ray supports a *unified and homogeneous approach* allowing to insert and retrieve XML-based data into, respectively from, an RDBS by providing an XML view over the relational context data, that may be accessed using XQuery. Second, *reuse of existing context schemata at the XML-side as well as at the RDBS-side* is possible, since the mapping knowledge mediates between independently developed schemata. Third, the *autonomy of the context schemata* is preserved, i.e., there is no need to change the context schemata for mapping purposes because of the flexible mapping possibilities provided by X-Ray. Fourth, *mapping of multiple schemata* at both sides is possible which means that a certain XML-based context schema can be mapped to multiple different database schemata and vice versa. Fifth, X-Ray enables *mapping transparency and storage transparency* when accessing context data, i.e., the application retrieving or updating context data does not need to have knowledge about the mapping and the storage schema – it simply works with the XML context data. This is since X-Ray provides an XML view over the context data stored within an RDBS. Sixth, X-Ray *eases the maintenance of the mapping knowledge* because of its reification within a meta schema. In addition, storing the mapping knowledge in a separate meta schema provides for a *loose coupling* with the schemata to be integrated, retaining their autonomy. Finally, *extensibility of the context* *model* is supported, since new XML schemata describing context data as well as new relational schemata can be easily introduced by simply inserting additional knowledge into the meta schema.

5 Conclusion

In this paper we have proposed an approach for managing XML-based context data by means of relational database technology. RDBS do not only allow to reuse existing legacy data for realizing context-aware services, but their features also suit well with the specific characteristics of context data, thus allowing efficient storage, retrieval, and processing. To enable the storage of context data possibly relying on heterogeneous XML schemata, within a relational schema, a middleware called X-Ray has been employed. X-Ray provides a generic mapping approach since the mapping knowledge is not hard-coded but rather reified within a meta schema. This enhances maintainability and changeability and provides the basis to automatically compose context data out of the RDBS when requested and decompose them when they have to be stored.

References

[1] J., Altmann, G., Leonhartsberger, M., Pichler, W., Schwinger, Th., Hofer, W., Retschitzegger: "Context-Awareness on Mobile Devices - the Hydrogen Approach", Proc. of the 36th Hawaii Int. Conf. on System Sciences (HICSS-36), Hawaii, January, 2003.

[2] B., Badrinath, et al.: "A conceptual framework for network and client adaptation", IEEE Mobile Networks and Applications (MONET), Vol. 5 No. 4, pp 221-231, 2000.

[3] E., Rahm, P.A., Bernstein: "A Survey of Approaches to Automatic Schema Matching", VLDB Journal 10, 4, December, 2001.

[4] P., Brézillon, J.-Ch., Pomerol: "Modeling and using context for system development: Lessons from experiences." Journal of Decision Systems. P. Humphreys, P. Brézillon (eds.), Vol. 10, No. 2, 2001.

[5] P., Brusilovsky, M.T., Maybury: "From adaptive hypermedia to adaptive Web", Communications of the ACM (CACM), Vol. 45, No. 5, P. Brusilovsky et al. (eds.), 2002.

[6] J.,M., Carroll, A.P., Aaronson: "Learning by Doing With Simulated Intelligent Help", Communications of the ACM (CACM), Vol. 31, No. 9, Sept. 1988.

[7] A. Deutsch, and V. Tannen, "MARS: A System for Publishing XML from Mixed and Redundant Storage", in Proc. of the 29th Int. Conference On Very Large Databases (VLDB), Berlin, 2003.

[8] A.K., Dey, D., Salber, G.,D., Abowd: "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications", special issue on Context-Aware Computing, Human-Computer Interaction (HCI) Journal, Vol. 16, No. 2-4, 2001.

[9] A. Eisenberg, and J. Melton, "SQL/XML is Making Good Progress", SIGMOD Record 31(2), 2002.

[10] M.F. Fernandez, W.-C. Tan, and D. Suciu, "SilkRoute: Trading between Relations and XML", in Proc. of the 9th Int. World Wide Web Conf. (WWW), Amsterdam, May 2000.

[11] A. Ferscha, S. Vogl, W. Beer: „Context Sensing, Aggregation, Representation and Exploitation in Wireless Networks", Future Generation Computing Systems, North Holland, (to appear), 2004.

[12] D. Florescu, and D. Kossmann, "Storing and Querying XML Data Using an RDBMS", IEEE Data Engineering Bulletin, Special Issue on XML, 22(3), Sept. 1999.

[13] M., J. Franklin, "Challenges in Ubiquitous Data Management", Informatics: 10 Years Back, 10 Years Ahead, R. Wilhiem (ed)., Springer LNCS No. 2000, 2001.

[14] J. Funderburk, G. Kiernan, J. Shanmugasundaram, E. Shekita, and C. Wei, "XTABLES: Bridging Relational Technology and XML", IBM Systems Journal 41(4), 2002.

[15] R. Goldman, et al.:, "From Semistructured Data to XML: Migrating the Lore Data Model and Query Language", Proc. of the 2nd Int. Workshop on WebDB, Philadelphia, June 1999.

[16] M. D. Good, J. A. Whiteside, D. R. Wixon, S. J. Jones: "Building a User-Derived Interface", Communications of the ACM (CACM), Vol. 27, No. 10, Oct. 1984.

[17] M., Hazas et al.: "Location-Aware Computing Comes of Age" IEEE Computer, Feb. 2004.
[18] U. Hohenstein, „Supporting XML in Oracle9i. XML Data Management“, Native XML and XML-Enable Database Systems, A.B. Chaudhri, et al. (eds.), Addison Wesley, 2003.
[19] IBM WebSphere Transcoding Publisher 4.0 Specification, http://www-306.ibm.com/software/pervasive/transcoding_publisher, 2004. [last access 2004-12-23]
[20] G. Kappel, E. Kapsammer, S. Rausch-Schott, and W. Retschitzegger, "X-Ray - Towards Integrating XML and Relational Database Systems", in Proc. of the 19th Int. Conf. on Conceptual Modeling (ER), LNCS 1920, Springer, Salt Lake City, USA, Oct. 2000.
[21] G., Kappel, W., Retschitzegger, E., Kimmerstorfer, W., Schwinger, Th., Hofer, B., Pröll: "Towards a Generic Customization Model for Ubiquitous Web Applications", Proc. of the 2nd Int. Workshop on Web Oriented Software Technology, Malaga, Spain, June 2002.
[22] G., Kappel, B., Pröll, W., Retschitzegger W., Schwinger: "Customization for Ubiquitous Web Applications - A Comparison of Approaches", Int. Journal of Web Engineering and Technology (IJWET), Inaugural Volume, Inderscience Publishers 2003.
[23] G. Kappel, E. Kapsammer, W. Retschitzegger: "Integrating XML and Relational Database Systems", World Wide Web Journal (WWWJ), Kluwer Academic Publishers, Vol. 7(4), December 2004.
[24] L. Kleinrock: "Nomadicity: Anytime, Anywhere In A Disconnected World", Mobile Networks and Applications, 1(4), 1999
[25] A. Kobsa, J. Koenemann, W. Pohl: "Personalized Hypermedia Presentation Techniques for Improving Online Customer Relationships", The Knowledge Engineering Review, Vol. 16, No. 2, 2001.
[26] S., Loeb, D., Terry: "Information Filtering", Communications of the ACM (CACM), 35(12),. 1992.
[27] R. Mohan, J. R. Smith, C-S. Li: "Adapting Multimedia Internet Content for Universal Access", IEEE Trans. on Multimedia, Vol. 1, No. 1, March 1999.
[28] M. Norrie, A. Palinginis: "Empowering Databases for Context-Dependent Information Delivery", Proc. of the Workshop on Ubiquitous Mobile Information and Collaboration Systems (UMICS), Klagenfurt/Velden, Austria, June 2003.
[29] T., Öszu, P., Valduriez: "Principles of Distributed Database Systems", Prentice-Hall, 1999.
[30] Open Geospatial Consortium (OpenGIS): "Geography Markup Language (GML), http://www.opengeospatial.org, [last access 2004-12-23]
[31] H. Schöning, J. Wäsch, „Tamino – An Internet Database System“,Proc. of 7th Int. Conf. on Ext. DB Technology (EDBT), Springer, LNCS 1777, Konstanz, March 2000.
[32] A., Schmidt, G., Kortuem, D., Morse, A., Dey, (Editors): "Situated Interaction and Context-Aware Computing, Personal and Ubiquitous Computing". Volume 5(1), 2001.
[33] D., Shasha, Ph., Bonnet: "Database Tuning: Principles, Experiments and Troubleshooting Techniques", Morgan Kaufman Publishing, 2002.
[34] K., Stolze: "SQL/MM Spatial - The Standard to Manage Spatial Data in a Relational Database System", GI-Fachtagung Datenbanksysteme für Business, Technologie und Web, Leipzig, Feb. 2003.
[35] K., Torp, S., Christian S., Jensen, R.,T. Snodgrass: "Effective Timestamping in Databases". Journal on Very Large Databases (VLDB). 8(3-4), 2000.
[36] G. Trajcevski, O., Wolfson, K., Hinrichs, S., Chamberlain: „Managing uncertainty in moving objects databases", ACM Transactions on Database Systems (TODS), Vol. 29, No. 3, 2004.
[37] Versant Software Corporation, www.versant.com, [last access 2004-12-23].
[38] G. Vossen, G. Weikum: "Transactional Information Systems", Morgan Kaufmann, 2001.
[39] P. Waddington, et al., "Oracle9i Application Server Wireless Edition in Action", Oracle Magazine, Jan. - Feb. 2002.
[40] Wireless Application Group, WAP-248-UAPROF-20011020-a, Wireless Application Protocol Forum, 20.10.2001.
[41] World Wide Web Consortium (W3C), "Composite Capabilities/Preference. Profiles", http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115, 2004. [last access 2004-12-23]
[42] World Wide Web Consortium (W3C), "Extensible Markup Language (XML) 1.0", W3C REC, Oct. 2000, http://www.w3.org/TR/2000/REC-xml-20001006, [last access 2004-12-23].
[43] World Wide Web Consortium (W3C), "XML Schema", W3C Recommendation, May 2001, http://www.w3.org/XML/Schema, [last access 2004-12-23].
[44] World Wide Web Consortium (W3C), "XQuery 1.0: An XML Query Language", W3C Working Draft, December 2004, http://www.w3.org/TR/xquery, [last access 2004-12-23]
[45] World Wide Web Consortium (W3C), "Platform for Privacy Preferences (P3P) Project", http://www.w3.org/P3P, [last access 2004-12-23].