Web data modeling for integration in data warehouses

Sami Miniaoui, Jérôme Darmont, Omar Boussaid
ERIC, Université Lumière Lyon 2
5 avenue Pierre Mendès-France
69676 Bron Cedex
France

Abstract *In a data warehousing process, the data preparation phase is crucial. Mastering this phase allows substantial gains in terms of time and performance when performing a multidimensional analysis or using data mining algorithms. Furthermore, a data warehouse can require external data. The web is a prevalent data source in this context, but the data broadcasted on this medium are very heterogeneous. We propose in this paper a UML conceptual model for a complex object representing a superclass of any useful data source (databases, plain texts, HTML and XML documents, images, sounds, video clips...). The translation into a logical model is achieved with XML, which helps integrating all these diverse, heterogeneous data into a unified format, and whose schema definition provides first-rate metadata in our data warehousing context. Moreover, we benefit from XML's flexibility, extensibility and from the richness of the semi-structured data model, but we are still able to later map XML documents into a database if more structuring is needed.*

Keywords: Multiform data, Data warehousing, Integration, Modeling

1 Introduction

In the context of e-commerce, analyzing the behavior of a customer, a product, or a company consists in monitoring one or several activities (commercial or medical pursuits, patent deposits, etc.). The objective of multidimensional analysis, particularly OLAP, is to analyze such activities under the form of numerical data. The information is summarized and can be presented as relevant information (i.e., knowledge), thus connecting OLAP to other analysis tools such as KDD (*Knowledge Discovery in Databases*) techniques (namely, data mining), whose objectives are to understand and predict the behavior of one or several activities.

To be efficient in terms of quality and response time, analysis tools need their input data to be properly structured, acquired, and prepared in a previous step. These data are typically stored in databases aimed at decision support (such as data warehouses) that we call Decision Databases (DDBs). These databases can necessitate external data sources. For instance, a company willing to support competitive monitoring cannot merely analyze only data from its own production databases. The web is a prevalent data source in this context. However, the data broadcasted on this medium are very heterogeneous, which makes their conceptualization in a data warehousing framework difficult. Nonetheless, the concepts of data warehousing [1] remain valid in this approach. Measures, though not necessarily numerical, remain the indicators for analysis, and analysis is still performed following different perspectives represented by dimensions. Large data volumes and their dating are other arguments in favor of this approach [2].

Our objective is to use the web as a full data source for DDBs, in a transparent way. This raises several issues:

- structuring multiform data from the web

— databases, plain texts, multimedia data (images, sounds, video clips...), semi-structured data (HTML, XML, or SGML documents) — into a database;

- integrating these data into the particular architecture of a data warehouse (fact tables, dimension tables, data marts);
- devising evolution strategies for the warehouse when new data pop up;
- physically reorganizing data depending on usage to improve query performance.

The aim of this paper is to address the first issue. We propose a unified UML model for a complex object representing a superclass of the multiform data we need to integrate in a DDB. Our objective is not only to store data, but also to truly prepare them for analysis. This is not a mere ETL (*Extracting, Transforming, and Loading*) task, which would only render data names and domains consistent.

We elected to translate our UML conceptual model into an XML [3] logical model, for several reasons. First, XML encapsulates both data and their schema, either implicitly or in a DTD. This representation is also found in data warehouses, which store both data and metadata that describe the data. Hence, XML is particularly adapted for our purposes. Moreover, we benefit from the flexibility, the extensibility and the richness of the semi-structured data model. And since XML documents can easily be mapped into a conventional (e.g., relational) database [4], we can also take advantage of well-structured data and query processing efficiency, if necessary when we move down to the physical model. Furthermore, XML-based databases like Lore [5] are quickly expanding. Hence, we get the best of both worlds (the structured and the semi-structured) by adopting the XML format.

The remainder of this paper is organized as follows. Section 2 establishes a short state of the art regarding XML mapping and federated, multimedia, and text databases. Section 3 presents our unified model for multiform data. Section 4 outlines how this conceptual model is translated into a logical, XML model. We finally conclude the paper and discuss future research issues in Section 5.

2 Related work

2.1 Data integration

We identified two main ways to integrate heterogeneous data into a data warehouse. The first approach relates to federated databases, that are distributed and heterogeneous databases constituted from data sources of various natures: HTML or XML documents, databases, and so on, and providing users with an integrated view of the data [6, 7]. The casual architecture for a federated database is layered in three levels:

- *presentation:* components allowing to formulate queries in the federated database language;
- *mediation:* mediators in charge of collecting queries issued by users in the presentation components and translating them in the proper language of each data source;
- *adaptation:* components allowing communication between data sources and mediators.

The second possible approach consists in capturing the common characteristics of the different data types we need to integrate. In order to propose a unified data model, we took interest in how data is structured, stored, and indexed in textual and multimedia databases. Indexing strategies in textual databases include reversed lists of significant terms with their frequency of appearance in each document, signatures obtained by hashing keywords, and relative frequency matrix of words present in a set of documents [8]. Multimedia databases may adopt the following characteristics to index images: signatures derived from (manually captured) keywords describing the image, color, texture or brightness distributions, etc. [9]

2.2 XML mapping

As XML emerged as a data exchange standard language, its storage in databases became a research issue. Several approaches have been adopted to map an XML document into a database. [10, 11] propose algorithms that exploit a UML schema to map a DTD into a relational schema. Another approach consists in representing an XML document as a labeled, oriented graph where vertices are data types, edges are classes or objects, and leaves are data [12]. In these two approaches, a relational or object-relational database system was used to store the XML documents, but XML-native DBMSs, such as LORE [5], also exist.

3 Unified conceptual model

The data types we consider (text, multimedia documents, relational views from databases) for integration in a data warehouse all bear characteristics that can be used for indexing. The UML class diagram shown in Figure 1 represents a complex object generalizing all these data types. Note that our goal here is to propose a general data structure: the list of attributes for each class in this diagram is willingly not exhaustive.

A complex object is characterized by its name and its source. The date attribute introduces the notion of successive versions and dating that is crucial in data warehouses. Each complex object is composed of several subdocuments. Each subdocument is identified by its name, its type, its size, and its location (i.e., its physical address). The document type (text, image, etc.) will be helpful later, when selecting an appropriate analysis tool (text mining tools are different from standard data mining tools, for instance). The language class is important for text mining and information retrieval purposes, since it characterizes both documents and keywords.

Eventually, keywords represent a semantic representation of a document. They are metadata describing the object to integrate (medical image, press article...) or its content. Keywords are essential in the indexing process that helps guaranteeing good performances at data retrieval time. Note that we consider only logical indexing here, and not physical issues rose by very large amounts of data, which are still quite open as far as we know. Keywords are typically manually captured, but it would be very interesting to mine them automatically with text mining, image mining, or XML mining (tag detection) techniques, for instance.

All the following classes are subclasses of the subdocument class. They represent the basic data types and/or documents we want to integrate. Text documents are subdivided into plain texts and tagged texts (namely HTML, XML, or SGML documents). Tagged text are further associated to a certain number of links. Since a web page may point to external data (other pages, images, multimedia data, files...), those links help relating these data to their refering page.

Relational views are actually extractions from any type of database (relational, object, object-relational — we suppose a view can be extracted whatever the data model) that will be materialized in the data warehouse. A relational view is a set of attributes (columns, classically characterized by their name and their domain) and a set of tuples (rows). At the intersection of tuples and attributes is a data value. In our model, these values appear as ordinal, but in practice they can be texts or BLOBs containing multimedia data. The query that helped building the view is also stored. Depending on the context, all the data can be stored, only the query and the intention (attribute definitions), or everything. For instance, it might be inadequate to duplicate huge amounts of data, especially if the data source is not regularly updated. On the other hand, if successive snapshots of an evolving view are needed, data will have to be stored.

Images may bear two types of attributes: some that are usually found in the image file header (format, compression rate, size in pixels, resolution), and some that need to be extracted by program, such as color or texture distributions.

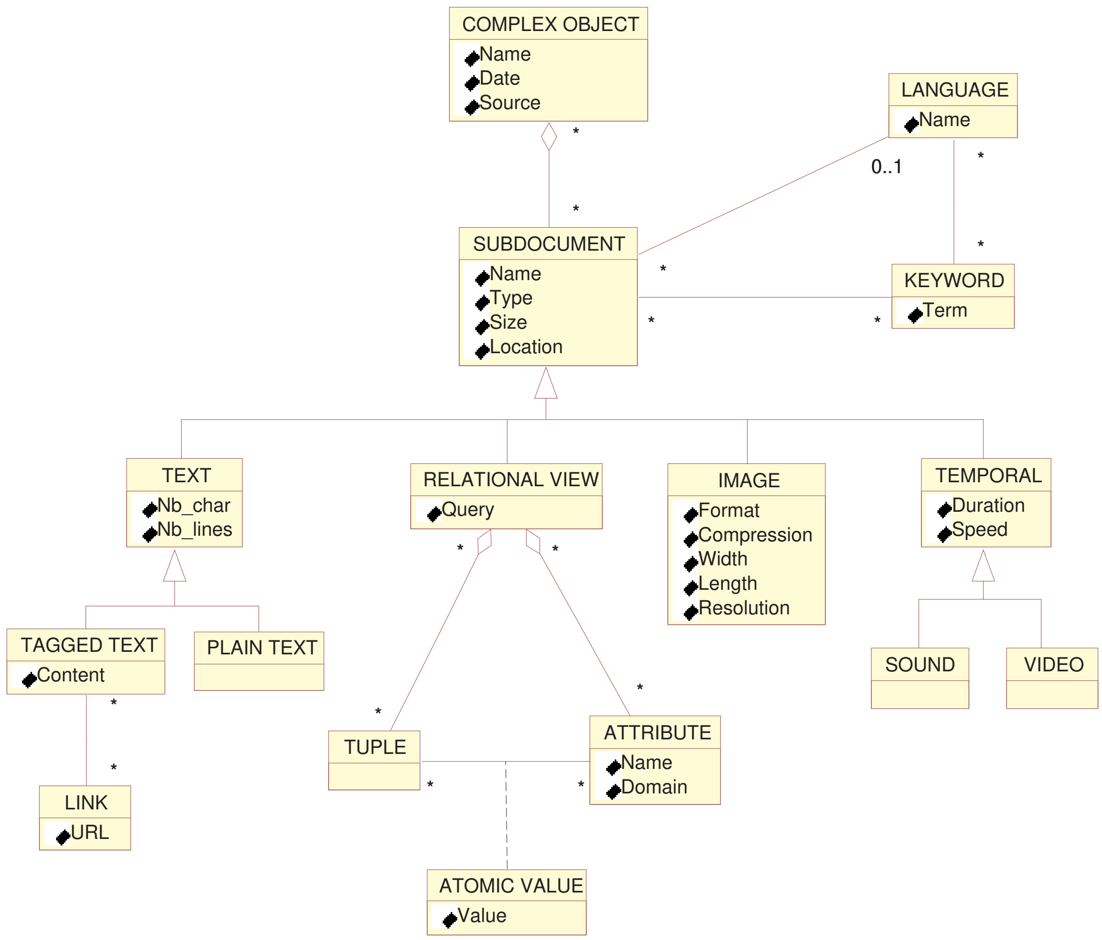

Figure 1: Multiform data model

Eventually, sounds and video clips are part of a same class because they share temporal attributes that are absent from the other types of data we consider. As far as we know, these types of data are not currently analyzed by mining algorithms, but they do contain knowledge. This is why we take them into account here (though in little detail), anticipating advances in "multimedia mining" techniques.

4 XML-based logical model

In a data warehouse, data come with metadata that describe their origin, the rules for transformations they may have undergone, and information regarding their usage [13]. Identically, multiform data modeled as complex objects may be viewed as *documents*. It is then necessary to consider two kinds of data: data themselves and sets of information pieces, such as descriptive data that allow their identification, or status data describing their semantics.

The use of XML as an implementation

tool for multiform data (i.e., complex objects) viewed as documents seems natural. This language indeed helps representing both the description and the content of any document. Within a modeling process, the translation of UML classes representing multiform data into XML constitutes a logical formalization phase. The obtained logical model can be as well mapped in a relational or object-relational database as stored in a native XML database.

4.1 XML DTD

When integrating multiform data, we adopt a classical information system modeling process: first devise a conceptual model, and then translate it into a logical model. The UML class diagram from Section 3 is our conceptual model. We consider XML as a fine candidate for logical modeling.

The UML model can indeed be directly translated into an XML DTD (*Document Definition Type*), as shown in Figure 2. However, we applied minor shortcuts not to overload the DTD. Since the *LANGUAGE*, *KEYWORD*, *LINK*, and *VALUE* classes only bear one attribute each, we mapped them to single XML elements, rather than having them be composed of another, single element. For instance, the *LANGUAGE* class became the *LANGUAGE* element, but this element is not further composed of the *Name* element. Eventually, since the *ATTRIBUTE* and the *TUPLE* elements share the same sub-element "attribute name", we labeled it *ATT_NAME* in the *ATTRIBUTE* element and *ATT_NAME_REF* (reference to an attribute name) in the *TUPLE* element to avoid any confusion or processing problem.

4.2 Transformation algorithm

We are currently in the process of developing a prototype capable of taking as input any data source from the web, fitting it in our model, and producing an XML document. The general algorithm for integrating multiform data in our unified model is provided in Figure 3. It exploits the DTD from Figure 2. The actual application is written in Java.

The principle of this algorithm is to browse the DTD, fetching the elements it describes, and to write them into the XML document, along with the associated values extracted from the original data, on the fly. Note that, when reading a DTD line, the current element we refer to is the one which is being described, e.g., *TEXT* in the `<!ELEMENT TEXT (NB_CHAR, NB_LINES, (PLAIN_TEXT | TAGGED_TEXT))>` DTD line. We also suppose that sub-elements are defined in the same order they are declared in their parent element. Missing values are currently treated by inserting an empty element, but strategies could be devised to solve this problem, either by prompting the user or automatically.

At this point, our prototype is able to process text, image, and multimedia data. Figures 4 and 5 illustrate how multiform data (namely, an SGML tagged text and an image) are transformed using our approach.

5 Conclusion and future issues

We presented in this paper a UML model for a complex object that generalizes the different multiform data that can be found on the web and that are interesting to integrate in a data warehouse as external data sources. Our model allow the unification of these different data into a single framework, for purposes of storage and, maybe more importantly, preparation for analysis. Data must indeed be properly "formatted" before OLAP or data mining techniques can apply to them.

Our UML conceptual model is then directly translated into an XML DTD and instantiated into an XML document, which we both view as part of a logical model in our (classical) modeling process. XML is the format of choice for both storing and describing the data. The DTD indeed represents the metadata. It is also very interesting because of its flexibility and extensibility, while allowing straight mapping

```
<!ELEMENT COMPLEX_OBJECT (OBJ_NAME, DATE, SOURCE, SUBDOCUMENT+)>
    <!ELEMENT OBJ_NAME PCDATA #REQUIRED>
    <!ELEMENT DATE PCDATA #REQUIRED>
    <!ELEMENT SOURCE PCDATA #REQUIRED>
    <!ELEMENT SUBDOCUMENT (DOC_NAME, TYPE, SIZE, LOCATION, LANGUAGE?, KEYWORD*, (TEXT |
RELATIONAL_VIEW | IMAGE | TEMPORAL))>
       <!ELEMENT DOC_NAME PCDATA #REQUIRED>
       <!ELEMENT TYPE PCDATA #REQUIRED>
       <!ELEMENT SIZE PCDATA #REQUIRED>
       <!ELEMENT LOCATION PCDATA #REQUIRED>
       <!ELEMENT LANGUAGE PCDATA #REQUIRED>
       <!ELEMENT KEYWORD PCDATA #REQUIRED>
       <!ELEMENT TEXT (NB_CHAR, NB_LINES, (PLAIN_TEXT | TAGGED_TEXT))>
          <!ELEMENT NB_CHAR PCDATA #IMPLIED>
          <!ELEMENT NB_LINES PCDATA #IMPLIED>
          <!ELEMENT PLAIN_TEXT PCDATA #REQUIRED>
          <!ELEMENT TAGGED_TEXT (CONTENT, LINK*)>
             <!ELEMENT CONTENT PCDATA #REQUIRED>
             <!ELEMENT LINK PCDATA #REQUIRED>
       <!ELEMENT RELATIONAL_VIEW (QUERY?, ATTRIBUTE+, TUPLE*)>
          <!ELEMENT QUERY PCDATA #REQUIRED>
          <!ELEMENT ATTRIBUTE (ATT_NAME, DOMAIN)>
             <!ELEMENT ATT_NAME PCDATA #REQUIRED>
             <!ELEMENT DOMAIN PCDATA #REQUIRED>
          <!ELEMENT TUPLE (ATT_NAME_REF, VALUE)+>
             <!ELEMENT ATT_NAME_REF PCDATA #REQUIRED>
             <!ELEMENT VALUE PCDATA #IMPLIED>
       <!ELEMENT IMAGE (COMPRESSION, FORMAT, RESOLUTION, LENGTH, WIDTH)>
          <!ELEMENT COMPRESSION PCDATA #IMPLIED>
          <!ELEMENT FORMAT PCDATA #IMPLIED>
          <!ELEMENT RESOLUTION PCDATA #IMPLIED>
          <!ELEMENT LENGTH PCDATA #IMPLIED>
          <!ELEMENT WIDTH PCDATA #IMPLIED>
       <!ELEMENT TEMPORAL (DURATION, SPEED, (SOUND | VIDEO))>
          <!ELEMENT DURATION PCDATA #IMPLIED >
          <!ELEMENT SPEED PCDATA #IMPLIED>
          <!ELEMENT SOUND PCDATA #IMPLIED>
          <!ELEMENT VIDEO PCDATA #IMPLIED>
```

Figure 2: XML DTD

```
// Initialization
Read DTD line
Stack root element
// Main loop
While stack not empty do
    Unstack element
    // Positioning on the current element description
    While element not found in the DTD and not EOF(DTD) do
       Read DTD line
    End while
    If element was found then
       For each value of the element do // For elements with + or * cardinality
          If element is atomic then
             Write elementBeginTag, elementValue, elementEndTag
          Else // Composite element
             Write elementBeginTag
             Stack element // Necessary to later write end tag
             For each sub-element (in reverse order) do
                If sub-element does not belong to a selection then
                // If element not in a list of the form (PLAIN_TEXT | TAGGED_TEXT)
                   Stack sub-element
                Else
                   If sub-element was selected then
                   // If the DTD document type matches the actual document type
                      Stack sub-element
                   End if
                End if
             End for
          End if
       End for
    Else
       Write elementEndTag // Close composite elements
    End if
End while
```

Figure 3: Multiform data integration algorithm

SGML document

```
<!DOCTYPE lewis SYSTEM "lewis.dtd">
<REUTERS TOPICS="YES" LEWISSPLIT="TRAIN"
CGISPLIT="TRAINING-SET" OLDID="12509"
NEWID="326">
<DATE>2-MAR-1987 06:41:06.17</DATE>
<PLACES><D>france</D></PLACES>
<COMPANIES>SNCF</COMPANIES>
<TEXT>
<TITLE>SNCF ISSUING THREE BILLION FRANC
DOMESTIC BOND</TITLE>
<DATELINE>PARIS, March 2</DATELINE>
<BODY>The French state railway company, the
Ste Nationale des Chemins de Fer Francaise
(SNCF), is issuing a three billion French
franc domestic bond in two tranches, the
bond issuing committee said.  Details of
the issue will be announced later and it
will be listed in the Official Bulletin
(BALO) of March 9.
The issue will be co-led by Banque Nationale
de Paris, Caisse Nationale de Credit Agricole
and the Societe Marseillaise de Credit.
REUTER</BODY> </TEXT>
</REUTERS>
```

XML model

```
<?XML version=1.0?>
<!DOCTYPE MultiformData SYSTEM "mlfd.dtd">
<COMPLEX_OBJECT>
 <NAME>Reuters Press Release</NAME>
 <DATE>May 15, 2001</DATE>
 <SOURCE>Reuters</SOURCE>
 <SUBDOCUMENT>
  <NAME>SGMLdoc</NAME>
  <TYPE>SGML</TYPE>
  <SIZE>820 Bytes</SIZE>
  <LOCATION>SGMLfile.sgml</LOCATION>
  <LANGUAGE>English</LANGUAGE>
  <KEYWORD>France</KEYWORD>
  <KEYWORD>SNCF</KEYWORD>
  <TEXT>
   <NB_CHAR>790</NB_CHAR>
   <NB_LINES>12</NB_LINES>
   <TAGGED_TEXT>
    <CONTENT>The document could be reproduced
here as a CDATA.</CONTENT>
   </TAGGED_TEXT>
  </TEXT>
 </SUBDOCUMENT>
</COMPLEX_OBJECT>
```

Figure 4: Sample logical model for a tagged text

into a more conventional database if strong structuring and retrieval efficiency are needed for analysis purposes.

Since this work is only at its premises, perspectives are numerous. The first, immediate task to make this work concrete is completing our prototype by making it capable of taking as input not only texts and multimedia data, but all the web data sources we identified in Figure 1. The problem of missing values also needs to be addressed in more details since we could reuse existing, automatic techniques.

In our next step, we will have to integrate the documents produced by our application into a data warehouse. We suppose that mapping the XML document into a relational or object-relational database will be straightforward using the techniques presented in [4].

Our XML modeling could also be improved by taking advantage of the features proposed in XML Schema [14] that are not supported by DTDs, such as typing and inheritance. In other respects, our XML formalization may also be considered as first-level logical modeling. A multidimensional representation with dimensions and facts would make up the second level and allow the warehousing of multiform data. This second modeling level has not been discussed in this paper, but it is one of our main goals for the integration of web data in a data warehouse.

Next, we do not envisage data mining as a front-end tool only. We believe integrating multi-form data in a data warehouse requires

Image	**XML model**
 User-prompted keywords: scissors, black, white	<?XML version=1.0?> <!DOCTYPE MultiformData SYSTEM "mlfd.dtd"> <COMPLEX_OBJECT> <NAME>Sample image</NAME> <DATE>2001-06-15</DATE> <SOURCE>Local</SOURCE> <SUBDOCUMENT> <NAME>Scissors</NAME> <TYPE>Image</TYPE> <SIZE>24694 Bytes</SIZE> <LOCATION>scissors.bmp</LOCATION> <KEYWORD>scissors</KEYWORD> <KEYWORD>black</KEYWORD> <KEYWORD>white</KEYWORD> <IMAGE> <FORMAT>Bitmap</FORMAT> <COMPRESSION>None</COMPRESSION> <WIDTH>256</WIDTH> <LENGTH>192</LENGTH> <RESOLUTION>100 dpi</RESOLUTION> </IMAGE> </SUBDOCUMENT> </COMPLEX_OBJECT>

Figure 5: Sample logical model for an image

more than a simple ETL: it requires intelligence. We plan to include intelligence into the very preparation of data by using data mining techniques to extract information that is useful for storage (indexing), multidimensional analysis or data mining itself. For instance, we could automatically build a list of pertinent keywords for a given document, whether it is a text or a multimedia file, or extract such characteristics as color, texture, and brightness distributions from an image. The pre-processing of these characteristics would save time for ulterior analysis. Clustering techniques could also be used as a new type of aggregation of multiform data in a context of multidimensional analysis.

Eventually, monitoring data usage could help physically reorganizing the data to enhance the data warehouse's response time. For instance, new clusters could be devised to improve the performance of multidimensional queries.

References

[1] S. Chaudhuri, U. Dayal. *An overview of data warehousing and OLAP technology. ACM SIGMOD*, 1997.

[2] R. Kimball, R. Mertz. *The Data Webhouse: Building the Web-enabled Data Warehouse.* John Wiley & Sons, 2000.

[3] T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler, eds. *Extensible Markup Language (XML) 1.0 (Second Edition).* W3C Recommendation, *http://www.w3.org/TR/2000/REC-xml-20001006*, Oct 2000.

[4] R. Anderson, M. Birbeck, M. Kay, S. Livingstone, B. Loesgen, D. Martin, S. Mohr, N. Ozu, B. Peat, J. Pinnock, P. Stark, K. Williams. *Professional XML Databases.* Wrox Press, 2000.

[5] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, J. Widom. *Lore: A Database Management System for Semi-structured Data. SIGMOD Record* 26(3):54–66, Sept 1997.

[6] S. Busse, R. Kutsche, U. Leser, H. Weber. *Federated Information systems: Concepts, Terminology and Architectures. Forschungsberichte des Fachbereichs Informatik*, 99(9), 1999.

[7] E. Bertino, B. Catania, G.P. Zarri. *Intelligent Database Systems.* Addison Wesley, 2001.

[8] G. Gardarin. *Internet et bases de données.* Eyrolles, 1999.

[9] T.C. Rakow, E.J. Neuhold, M. Löhr. *Multimedia Database Systems – The Notions and the Issues. Datenbanksysteme in Büro, Technik und Wissenschaft BTW*, GI-Fachtagung, Dresden, 1–29, 1995.

[10] G. Booch, M. Christerson, M. Fuuchs, J. Koistinen. *UML for XML schema mapping specification. Technical report*, Rational Software, 1999.

[11] G. Kappel, E. Kapsammer, W. Retschitzegger. *X-Ray – Towards Integrating XML and Relational Database Systems. 19th International Conference on Conceptual Modeling*, 339–353, 2000

[12] G. Gardarin, F. Sha, T.D. Ngoc. *XML-based components for federating multiple heterogeneous data sources. LNCS* 1728:506–519, 1999.

[13] M.C. Wu, A.P. Buchmann. *Research Issues in Data Warehousing. BTW '97*, Ulm, Mar 1997.

[14] D.C. Fallside, ed. *XML Schema.* W3C Recommendation, *http://www.w3.org/TR/xmlschema-0/*, May 2001.