Integration of weakly heterogeneous semistructured data

George Feuerlicht[1 3] & Jaroslav Pokorný[2] & Karel Richta[2] & Narongdech Ruttananontsatean[1]

[1] University of Technology, Sydney, Australia, {jir@it.uts.edu.au}

[2] Charles University of Prague, Prague, Czech Republic, {jaroslav.pokorny@mff.cuni.cz, karel.richta@mff.cuni.cz}

[3] University of Economics, Prague

Abstract: While most business applications typically operate on structured data that can be effectively managed using relational databases, some applications use more complex semistructured data that lacks a stable schema. XML techniques are available for the management of semistructured data, but such techniques tend to be ineffective when applied to large amounts of heterogeneous data, in particular in applications with complex query requirements. In this paper we describe an approach that relies on the mapping of multiple semistructured data sets to object-relational structures and uses an object-relational database to support complex query requirements. We illustrate this approach using weakly heterogeneous oceanographic data.

Keywords: Weakly heterogeneous data, semistructured data, data integration

1. Introduction

Many organizations are facing the challenge of managing large amounts of complex information stored in different databases and files across multiple computer systems. This situation is particularly difficult to address in environments where traditional, structured data is combined with *semistructured* data, as such data is irregular and often incomplete making it difficult to manage using traditional database technology. While data management solutions exist for each particular type of data (i.e. structured, unstructured, and semistructured) in practice it is difficult to manage such data in an integrated way (Buneman, 1997).

Semistructured data is commonly encountered in various business applications including document processing systems, web-based applications, and in scientific data-intensive environments. Because of the lack of uniformity (each data set tends to have a slightly different structure) it is not possible to describe semistructured data using a database schema and consequently self-describing formats are widely used to manage this type of information. XML (W3C, 2004) is now being

used almost universally as a standard formatting language for the description of semistructured data.

However, the lack of the uniformity across different data sets and the heterogeneity of storage formats and data models (i.e. relational databases, XML files, etc.) make the data difficult to manage. Addressing this problem by embedding schema information within the data records using a markup language does not, in itself, provide a mechanism for managing large amounts of heterogeneous, semistructured data in an efficient manner. Because of the increasing importance of semistructured data to many organizations today, extensive research is being conducted to devise efficient methods to manage such data (Rahayu, 2007).

In this paper we focus on a particular type of semistructured data that we call *weakly heterogeneous*. Weakly heterogeneous data is characterized by relatively small variations in the structure between individual records, caused for example by changes in the data collection method (e.g. collecting additional attributes during patient follow up visits). This type of data is common in many business and scientific applications, for example in business portals and healthcare applications, where individual records have slightly different data structure that may include additional data elements, and possibly omit some existing data elements.

The main contribution of this paper is to describe a technique for the integration of weakly heterogeneous XML data sets that is suitable for the analysis of business and scientific information. The approach is based on the type apparatus of objects in an object-relational database and there is no requirement on existence of a schema for semistructured data to be integrated. We illustrate our approach using an example of oceanographic data and introduce the concept of a *disposable* data warehouse - a segment of the data stored temporally in an object-relational database and structured for the purposes of the analysis. The data warehouse supports repeated queries over a specific segment of weakly heterogeneous data, and is discarded once the analysis is complete.

The rest of the paper is organized as follows. In section 2 we discuss the challenges of managing weakly heterogeneous data using an illustrative example of oceanographic information. In section 3 we review related research, and in section 4 we describe our approach to integrating weakly heterogeneous data sets. In the final section, (section 5) we give our conclusions and briefly outline future work.

2. Oceanographic Data Example

To illustrate the challenges of managing weakly heterogeneous data, consider an oceanographic data scenario. Organizations that are responsible for the collection and management of oceanographic data such as the Australian Oceanographic Data Center (AODC: http://www.aodc.gov.au/) have to deal with a large collection of scientific data stored in a variety of different formats. In addition to data collected in Australia, AODC exchanges data sets with other organizations such as

the U.K. National Oceanography Centre (http://www.noc.soton.ac.uk/). The primary functions of AODC is to collect oceanographic data from local and international sources and deliver oceanographic information to organizations that request various subsets of the data for research purposes. In order to service such requests AODC needs to provide an effective query support across a large and complex collection of oceanographic data. What makes this particularly challenging is the nature of oceanographic information. Oceanographic data is characterized by:

i) Very large data volumes collected over a long period of time; the total size of active data is measured in terabytes.
ii) Complexity of the data structures; the data sets typically contain series of measurements (e.g. water temperature and salinity) measured at a given location and point in time (i.e. data has spatial and time coordinates).
iii) Data sets are irregular as a result of data being acquired using different instrumentation and measuring techniques over a period of time; new measuring techniques and instrumentation typically result in additional data items.
iv) Data is *sparse*, i.e. measurements only exist for some spatial coordinates and points in time.
v) Data is heterogeneous; i.e. data is stored in a multitude of data formats and databases, potentially associated with different semantics of the information.

Furthermore, new instrumentation typically produces better quality measurements, so that the accuracy of data varies over time.

```
<?xml version="1.0"?>
<marineData>
<marineRecord id="24" ver="1" cavCode="0">
<obsHeader>
  <obsID>689</obsID>
  <srcAgency>AODC</srcAgency>      ...
  <obsDate year="2000" month="6" day="23" hour="9" min="14"/>
  <latitude deg="12" min="11" sec="42" hem="S">-12.195</latitude>
  <longitude deg="130" min="11" sec="42"
           hem="E">130.195</longitude> ...
</obsHeader>
<ancillaryObservations>
  <ancObs aid="1" . . . unitsCode="DEGC" QCF="0">24.5</ancObs>
  <ancObs aid="2" . . . unitsCode="METR" QCF="0">72.0</ancObs>
</ancillaryObservations>
<obsData>
  <profile pid="1">
    <profileHeader>
      <dataType>TEMP</dataType> ...
      <obsUnits>DEGC</obsUnits><numObs>705</numObs> ...
    </profileHeader>
    <profileFlags><pFlag z="46.72">1</pFlag></profileFlags>
    <profileData><param ID="0" z="0.66" QCF="0">25.9</param> ...
  </profile>
</obsData>
</marineRecord>
           ...
</marineData>
```

Figure 1: Example of oceanographic temperature profile data (marineData.xml)

Figure 1 shows a typical example of oceanographic temperature profile data. The data contains a range of measured parameters, including the ocean water tempera-

ture at various levels of depths. Each temperature profile record consists of information that describes the context of the measurement, including the geographic position and the date of measurement, the instruments used, surface conditions, and a series of the temperature observations. Measurements can be taken from a stationary or moving platform (e.g. a ship), using various types of instruments. While all temperature profile data sets record essentially the same information (i.e. water temperature at various levels of depth), there are significant differences in the measured parameters, the number of observations collected, the accuracy of the measured values, and other attributes associated with the measurements. Although there are attempts to standardize oceanographic data, see for example (Isenor, 2005), most existing data sets exhibit variations in data structures and can be characterized as weakly heterogeneous data.

2.1 Querying Weakly Heterogeneous Data

Existing XML query languages such as XQuery (Boag, 2005) can be used to query the temperature profile data. For example, the query (Q1) "*List temperature records for latitude between -15 and 3*", can be written in XQuery as shown in Figure 2, with the corresponding result fragment shown in Figure 3:

```
<marineData>
  { FOR $m IN doc('marinedata.xml')/marineData/marineRecord
    LET $n := $m//param
    WHERE $m//latitude > -15 and $m//latitude < 3
    RETURN <temperature>{data($n)}</temperature> }
</marineData>
```

Figure 2: Example query Q1 using XQuery

Figure 3 shows the corresponding result set:

```
<marineData>
   <temperature>25.9</temperature>
   <temperature>25.82</temperature> …
</marineData>
```

Figure 3: Q1 result fragment in XML

However, because of the large amounts and complexity of such data, XML query techniques cannot support user requirements in an effective manner. Consider, for example, another typical query (Q2) "*Find the maximum temperature for latitudes between -15 and 3, longitudes between 100 and 140, at the depth between 100 and 200 meters, for the month of June, during the period of 2000 to 2007*". Evaluating this type of query requires spatial query support, and typically involves combining several temperature profile data sets with potentially different data structure (e.g. the level of depths at which temperature is measured can vary between records).

An important aspect of the operation of an oceanographic data center is the delivery of data sets to clients (on-demand) for further analysis. This type of analysis typically involves repeated queries over a pre-specified segment of the data, for example Australian temperature profile data collected between 2000 and 2007.

While the extraction of this data set will typically involve the integration of a number of weakly heterogeneous data records (i.e. individual temperature profiles), and is best done using XML query techniques (e.g. XQuery), the subsequent analysis of the data that involves repeated queries over this data set needs to be supported with a database management systems with sophisticated query capabilities that include spatial query support.

Our proposal is to re-structure the extracted data segment integrating individual data sets, and store the data in an object-relational database in a form suitable for analysis. The resulting data warehouse of oceanographic information is made available to the users who can execute repeated queries over this data set using the full capabilities of an ORDBMS (Object-Relational Database Management System). This solution involves a combination of XML and object-relational database techniques. We first generate an object-relational schema by traversing XML data from multiple temperature profile data sets, and store the resulting data segment in an object-relational database (Oracle 10g). The users can then execute repeated queries against an SQL 2003 compatible object-relational database. Results of the queries are returned as XML documents or in another specified format (e.g. HTML).

3. Related Research

XML data formats are widely used in applications such as Web portals and other document-intensive environments to store, interchange and manage semistructured data. These activities are made difficult by the issues discussed in the previous section that include multiple data formats and lack of metadata standardization. Using XML-based systems to manage this kind of information provides a number of advantages such as dynamic and flexible presentation, and enhanced search capabilities (Suresh, 2000). There is also a trend towards industry-wide standardization of XML formats based on industry specific markup language standards such as the Geography Markup Language (OCG, 2008), addressing the problem of data heterogeneity.

There are many different storage options for XML data with corresponding query capabilities. Depending on the application constraints one may choose a file system, a native storage system, see for example (Fiebig, 2002), (Harold, 2005), or a XML-enabled ORDBMS (Mlýnková, 2005).

Native XML databases provide storage for XML documents in their native format together with support for the search and retrieval of these documents (see (Bourret, 2007) for a list of available commercial products). Object-Relational DBMSs such as Oracle 10g provide a repository functionality called XML native type to store XML documents in the database without any conversion, and support updates, indexing, search, and multiple views on this data type (Liu, 2005). As an

alternative to storing data in the native XML format, XML data can be *shredded* (i.e. converted) into relational (or object-relational) tables (Beyer, 2005).

Other studies (Shanmugasundaram, 2001) propose alternative ways of efficiently constructing fully materialized XML views of the data using SQL extended with element constructors to allow mapping to XML, and at the same time supporting XML query translation to SQL. Both approaches, i.e. native XML and conversion to object-relational structures, have advantages and drawbacks. Querying semistructured data directly in the native XML format has the advantage of avoiding the need for conversion, but does not have the benefits of query optimization and advanced query capabilities (e.g. spatial query support) available in ORDBMSs, and may require introduction of additional features such as numbering schemes.

Surprisingly, most XML-enabled approaches do not use type apparatus of objects, or suppose an existence of a schema of semistructured data expressed in DTD or XML Schema language; see, for example (Runapongsa, 2002), (Amornsinlaphachai, 2006).

The above reviewed approaches typically address the management of semistructured data in general without paying specific attention to the type and extent of data heterogeneity, and may not be suitable in situations where intensive analysis of a specific segment of the data derived from multiple heterogeneous data sets is required. Our approach, described in the next section, combines the use of an XML query language for the integration of multiple weakly heterogeneous XML data sets and the capabilities of object-relational databases following the transformation of the XML data into an object-relational form. Another benefit of this approach is that the semistructured data can now be combined with other data held in the object-relational database.

4. Proposed Approach

Commercially available object-relational technology provides a comprehensive solution for the management of XML data. This allows the storage of XML data in the database, either fully or partially. A combination of storing XML data externally and internally within the database may be appropriate in situations where the data volumes are very large and access to the entire set of data at the same time is not required, as is the case in the previously described AODC environment. In order to address the requirements of applications that use weakly heterogeneous data (i.e. data sets with *slightly* varying structure) several steps are required:

i) integration of weakly heterogeneous XML data sets,
ii) extraction of object-relational schema from XML data to be integrated, and
iii) transformation of XML data and its storage in the ORDBMS.

4.1 Query system architecture

Figure 4 describes the proposed system architecture and the steps required to generate the data warehouse to support repeated queries over a selected data segment. First, given the user data requirements specified via the user interface a selected data segment, e.g. Australian temperature profiles for the period 2000-2007, is extracted from the corresponding XML datasets (step 1).

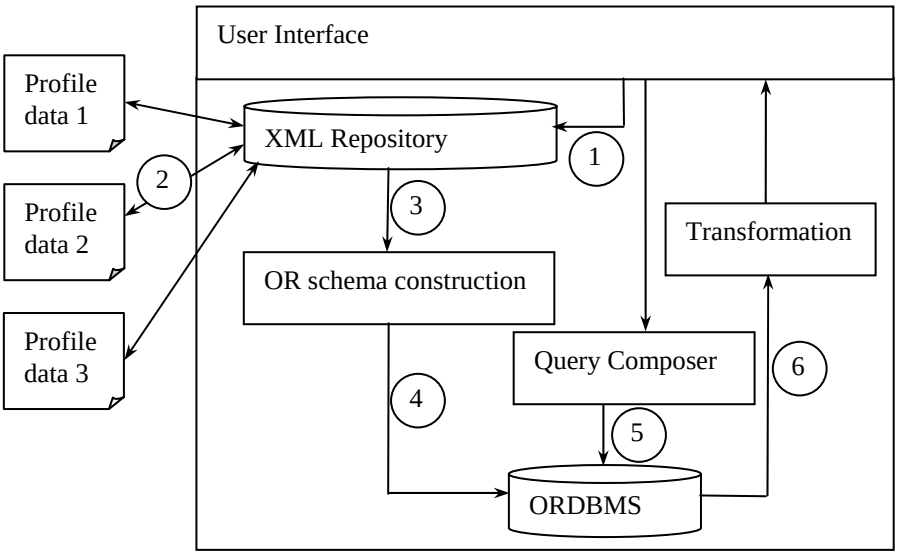

Figure 4: Query System Architecture

The system extracts this data by identifying the relevant XML files (individual temperature profiles, in this example) using the XML Repository. The XML Repository holds information such as the date and the position where the experiment was conducted. The identified XML files are then processed and using XQuery, aggregating the results into a single XML document (step 2). The corresponding object-relational schema is then generated (step 3), and the XML data is mapped into object-relational structures and stored in the ORDBMS (step 4). At this point the data warehouse constructed from weakly heterogeneous XML data sets is ready to support user queries.

The Query Composer is used to rewrite user queries presented via a graphical interface into SQL and the query is executed against the ORDBMS (step 5). The query results can be published as an XML document or transformed (step 6) into another presentation format (e.g. plain text, HTML etc.), as required by the user.

This system architecture and the implementation of a proof-of-concept prototype are described in detail elsewhere (Ruttananontsatean, 2007). In this paper we focus on extracting object-relational schema from XML data sets (i.e. step 3 in Figure 4).

4.2 Mapping schema

The oceanographic data sets that we use to illustrate our approach are characterized by deeply nested data structures that make the data unsuitable for mapping into *flat* relational tables. We use object-relational mapping that supports nested structures and fits well with this type of data. We note here that there is no direct unique mapping between XML structures and an object-relational schema. There are a number of alternative methods that can be used to perform the mapping. One approach, for example, is to create an object-relational view over the target schema, and to make the view the new query target. Another possibility is to use XSLT (Clark, 1999) to transform the XML document (i.e. to transform XML attributes to elements, etc.). We use the latter approach to *normalize* the original XML document so that it can be mapped to object-relational structures. We apply a well-known principle for the transformation - an element:

```
<E attr1=a1 attr2=a2 …attrk=ak>E-content</E>
```

is transformed to:

```
<E>
  <attr1>a1</attr1>
  <attr2>a2</attr2>
    ...
  <attrk>ak</attrk>
  <value>E-content</value>
</E>
```

in a situation where `E` is a leaf element. Otherwise, `E-content` replaces the `value` element. Figure 5 shows a general XSLT template for transforming the temperature profile data into a new temperature profile data (XML format).

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="*">
    <xsl:copy><xsl:for-each select="@*">
        <xsl:element name="{name()}">
        <xsl:value-of select="."/></xsl:element></xsl:for-each>
      <xsl:apply-templates select="./*"/>
      <xsl:choose><xsl:when test="count(@*) > 0">
          <xsl:element name="value"><xsl:value-of select="."/>
          </xsl:element></xsl:when>
        <xsl:otherwise><xsl:if test="string-length(text())> 0">
          <xsl:value-of select="."/></xsl:if></xsl:otherwise>
      </xsl:choose>
    </xsl:copy>
  </xsl:template>
</xsl:stylesheet>
```

Figure 5: XSLT template (skeleton)

Applying the template in Figure 5 to the temperature profile data shown in Figure 1 produces the transformation shown in Figure 6. We now proceed to devise an algorithm to generate an object-relational schema that conforms to the SQL standard (ISO, 2003) from the normalized data sets.

The transformation algorithm generates user-defined types, more specifically structured types. Structured types that we need to use for this purpose have an internal structure composed from other structured types and include collections (i.e. `ARRAY`s, `MULTISET`s). The definition of the basic data types have to be added

manually. We use the Oracle SQL syntax for our examples (we note that Oracle implements a subset of the SQL 2003 standard).

```
<?xml version="1.0"?>
<marineData>
 <marineRecord>
   <id>24</id> ...
   <obsHeader> ...
     <latitude>
       <deg>12</deg>
       <min>11</min>
       <sec>42</sec>
       <hem>0</hem>
       <value>-12.195</value>
     </latitude>
     <longitude> ...
       <hem>0</hem>
       <value>130.195</value>
     </longitude>      ...
   </obsHeader>
   <ancillaryObservations>
     <ancObs>
       <aid>1</aid> ...
   <unitsCode>DEGC</unitsCode>
       <value>24.5</value>
     </ancObs>
     <ancObs>
       <aid>2</aid>   ...
 </marineRecord>
     ...
</marineData>
```

Figure 6: Temperature profile data

The algorithm assumes a DOM version of a normalized XML document D. The associated tree is traversed in a depth-first manner to extract information about elements and their structure. Due to normalization of XML data we will not consider the order of elements. Consequently, the order of attributes of a structured type T is not relevant, and we consider T to be a set of attributes, and denote it as AT. The key principle of our transformation is to assign an SQL structured type to each element of XML data in the repository. Text nodes will become attributes of a basic type, repeating subelements of an element will compose a collection (e.g. ARRAY, MULTISET).

Formally, we consider a structured type $T(T_1, \ldots, T_n)$ whose construction depends on recognized occurrences of elements in the traversal and uses the following rules for generating the types:

1. IF E is the element associated with the first visit a node in the traversal and T is its associated type to be generated THEN $A_T := \emptyset$;
2. IF E_i is a direct subelement of E and T_i its associated type. THEN IF $T_i \in A_T$
   THEN generate collection type T_{Ti} whose members are of type T_i;
   $A_T := (A_T - T_i) \cup T_{Ti}$
   ELSE IF $T_{Ti} \notin A_T$ THEN $A_T := A_T \cup T_i$;
3. IF no further E_i exists and $A_T \neq \emptyset$ THEN generate T;

Since the XML data appears as a collection of specific records, we can stop generating types before the last generated collection type T_{Ti}. Then the database objects can be defined by `CREATE TABLE DB-`T_i `OF` T_i statement.

A problem can arise when type *T* that was already generated earlier during the traversal appears again with potentially different semantics. This may require user intervention to align the semantics of the generated types. A comprehensive discussion of the resolution of such semantic conflicts can be found in (Ruttananontsatean, 2007) and will not be considered further in this paper.

In our implementation we generate names of types for each tag `tag` as `tagType`. This avoids problems with missing elements, as all elements have to be mapped to nullable columns - the default for every nonprime attribute of a relation. Figure 7 shows a fragment of database schema in Oracle 10g syntax generated for data in Figure 6.

```
CREATE TYPE ancObsType AS OBJECT
(           aid          NUMBER,
            paramCode    VARCHAR2(30),
            unitsCode    VARCHAR2(30),
            QCF          NUMBER,
            value        NUMBER(20,10) );
CREATE TYPE ancillaryObservationsType
AS VARRAY(1000) OF ancObsType; …
CREATE TYPE marineRecordType AS OBJECT
(           id           NUMBER,
            ver          NUMBER,
            cavCode      NUMBER,
            obsHeader    obsHeaderType, …
            obsData         obsDataType);
CREATE TABLE marineRecord OF marineRecordType;
```

Figure 7: Object-Relational Schema (a fragment)

One potential improvement of the algorithm concerns the case when two structured types are differentiated only by name (i.e. are synonymous). Then the set of attributes can be aggregated into one type as shown in Figure 8. The name of such type can be generated or determined manually.

```
CREATE coordinateType AS OBJECT
(           deg NUMBER,
            min NUMBER,
            sec NUMBER,
            hem VARCHAR2(30),
            value NUMBER(20,10) );
CREATE obsHeaderType AS OBJECT
(           obsID  NUMBER,
            …
            latitude   coordinateType,
            longitude  coordinateType, …);
```

Figure 8: Object-Relational Schema (a fragment)

The example database uses a single table of oceanographic data with richly structured rows.

5. Conclusions and Future Work

In this paper we describe an approach suitable for supporting complex query requirements of applications that use semistructured data. We have described the overall system architecture and focused on the transformation of weakly heterogeneous data sets into the corresponding object-relational structures. We have illustrated our approach using weakly heterogeneous oceanographic data sets that exhibit complex multi-dimensional data. We note, however, that such weakly

heterogeneous data is common in many business applications (e.g. Web portals), data warehousing applications and in application domains such as healthcare. Frequently, such data needs to be subject to intensive analysis and often involves multi-dimensional data that requires the support of object-relational database management system, making the proposed approach of general interest.

Further work is needed to address the various semantic issues that typically arise in environments that use heterogeneous semistructured data to assist the users in semi-automatic resolution of semantic conflicts.

Acknowledgments

This research has been partially supported by the National Program of Research, Information Society Project No. 1ET100300419 and also by the grant of GACR No. GA201/06/0756 a GACR No. GA201/06/0175.

References

Amornsinlaphachai, P., Rossiter, N. & Ali, M. A. (2006) Storing Linked XML Documents in Object-Relational DBMS. *Journal of Computing and Information Technology (CIT)*, vol.14, no.3, pp.225–241.

Beyer, K., Cochrane, R. J., Josifovski, V., Kleewein, J., Lapis, G., Lohman, G., Lyle, B., Ozcan, F., Pirahesh, H., Seemann, N., Truong, T., Van der Linden, B., Vickery, B. & Zhang1, Ch. (2005): *System RX: One Part Relational, One Part XML*. In *Proc. of the 2005 ACM SIGMOD Int. Conf. on Management of Data, Baltimore*, Maryland, USA, ACM Press, pp. 347–358.

Boag, S., Chamberlin, D., Fernández, M. F., Florescu, D., Robie, J. & Siméon, J. (2005) XQuery 1.0: An XML Query Language, W3C Working Draft, 04 April 2005. Retrieved July 4, 2008 from: http://www.w3.org/TR/xquery/

Bourret, R. (2007) XML Database Products. Retrieved July 4, 2008 from: http://www.rpbourret.com/xml/XMLDatabaseProds.htm

Buneman, P. (1997) *Semistructured Data*. In *Proc. of 1997 Symposium on Principles of Database Systems (PODS97)*, Tucson, Arizona, pp.117-121.

Clark, J. (1999): XSL Transformations (XSLT) Version 1.0. W3C Recomm. Nov 16, 1999.

Fiebig, T., Helmer, S., Kanne, C.-C., Moerkotte, G., Neumann, J., Schiele, R. & Westmann, T. (2002): Anatomy of a native XML base management system. *VLDB Journal*, 11(4): pp.292–314.

OCG (2008) Geography Markup Language. Retrieved July 4, 2008 from: http://www.opengeospatial.org/standards/gml

Harold, E.R. (2005) Managing XML data: Native XML databases. Retrieved July 4, 2008 from: http://www.ibm.com/developerworks/xml/library/x-mxd4.html

Isenor, A.W. & Keeley, J.R (2005) Modeling Generic Oceanographic Data Objects in XML. *Computing in Science & Engineering*, July/August, pp.58-65.

ISO/IEC 9075:2003 (2003) Information Technology, Database Languages, SQL. Part 2: Foundations.

Liu, Z. H., Krishnaprasad, M. & Arora, V. (2005): *Native XQuery Processing in Oracle XMLDB.* In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, Baltimore, Maryland, pp. 828-833.

Mlýnková, I. & Pokorný, J. (2005) XML in the World of (Object-)Relational Database Systems. In Vasilecas, O.; et al (Eds.), *Information Systems Development Advances in Theory, Practice, and Education 2004*. Springer Science+Business Media, Inc., pp.63-76.

Rahayu, J.W., Pardede, E. & Taniar, D. (2007) XML Databases: Trends, Issues, and Future Research. In *The 9th Int. Conf. on Information Integration and Web-based Applications and Services (iiWAS 2007)*, ACS, vol.229, pp.9-10, Jakarta, Indonesia. http://www.ocg.at/publikationen/books/volumes/sr229.html

Runapongsa, K. and Patel, J.M. (2002) Storing and Querying XML Data in Object-Relational DBMSs. In *XML-Based Data Management and Multimedia Engineering*. EDBT 2002 Workshops, LNCS 2490/2002.

Ruttananontsatean, N. (2007) An Investigation of Query Techniques for Semistructured Data, PhD Thesis, Faculty of Information Technology, University of Technology, Sydney, Australia, November 2007.

Shanmugasundaram, J., Shekita, E., Barr, R., Carey, M., Lindsay, B., Pirahesh H. & Reinwald B. (2001) Efficiently Publishing Relational Data as XML Documents. *VLDB Journal* 10(2-3): pp.133-154.

Suresh, R., Shukla, P. & Schwenke G. (2000) XML-Based Data Systems for Earth Science Applications. In *Proc. of Geoscience and Remote Sensing Symposiun (IGARSS)*, IEEE 2000 Int., vol.3, pp.1214-1216.

W3C (2004): Extensible Markup Language (XML) 1.1. (Second Edition), W3C Recommendation 16 August 2004. Retrieved July 4, 2008 from: http://www.w3.org/TR/xml11/