# Integrating XML and Databases

**Elisa Bertino** • *University of Milano, Italy* • *bertino@dsi.unimi.it*
**Barbara Catania** • *University of Genova, Italy* • *catania@disi.unige.it*

XML is becoming a standard for data communication over the Internet. Like HTML, it is a markup language, but it supports a richer set of features, such as user-defined tags that allow both data and descriptive information about data to be represented within a single document. At the same time, presentation aspects remain decoupled from data representation. XML's flexibility lets it serve as a metalanguage for defining other markup languages specialized for specific contexts. A document type definition (DTD) describes the tags documents can use, customized to the specific semantic requirements of the application context, and the rules connecting tags with their contents. These capabilities make XML a common data format for data interchange between computer systems and between applications.

XML's proliferation raises the question of how data transferred by XML documents can be read, stored, and queried. In other words, how can database management systems (DBMSs) handle XML documents?

## Classifying XML Documents

As R. Bourret points out,[1] we can look at XML documents in two ways. In one view they serve as containers for data transferred over the Web. In this sense they are data-centric, since their meaning depends only on the structured data represented inside them. Typical data-centric documents are regular in structure and homogeneous in content. They occur in business-to-business applications such as buyer-supplier trading automation, inventory database access and sharing, integration of commercial transactions, and workflow. The data-centric XML document represented in Figure 1 contains invoice information.

In another view, XML documents are application-relevant objects — that is, new data objects to be stored and managed by a DBMS. In this sense they are document-centric, since their meaning depends on the document as a whole. Their structure is more irregular, and data are heterogeneous. Examples include books, e-mail messages, and any XHTML document (a combination of XML and HTML that models both structure and presentation layout). This view of XML documents prevails in personalized publishing, portals, customized presentations, and content and document management applications.

Figure 2 represents a document-centric document taken from "XML and Databases."[1]

```
<Invoices>
     <Invoice Number = "12">
          <Seller SellerNum = "32">
               <SellerName> XYZ Inc. </SellerName>
               <Address> ABC Street, New York </Address>
          </Seller>
          <InvoiceDate> 12/5/01 </InvoiceDate>
          <InvoiceLine>
               <Product> Product 1 </Product>
               <Quantity> 3 </Quantity>
               <UnitPrice> 1000 </UnitPrice>
          </InvoiceLine>
          <InvoiceLine>
               <Product> Product 31 </Product>
               <Quantity> 2 </Quantity>
               <UnitPrice> 1500 </UnitPrice>
          </InvoiceLine>
     </Invoice>
</Invoices>
```

*Figure 1. A data-centric XML document. Its data structure is quite regular, and the content of the various tags is homogeneous.*

```
<Product>

<Name>Turkey Wrench</Name>

<Developer>Full Fabrication Labs, Inc.</Developer>

<Summary>Like a monkey wrench, but not as big.</Summary>

<Description>

<Para>The turkey wrench, which comes in both right- and left-handed versions (skyhook optional),
is made of the finest stainless steel. The Readi-grip rubberized handle quickly adapts to your
hands, even in the greasiest situations. Adjustment is possible through a variety of custom
dials.</Para>

<Para>You can:</Para>

<List>
<Item><Link URL="Order.html">Order your own turkey wrench</Link></Item>
<Item><Link URL="Wrenches.htm">Read more about wrenches</Link></Item>
<Item><Link URL="catalog.zip">Download the catalog</Link></Item>
</List>

<Para>The turkey wrench costs just $19.99 and, if you order now, comes with a hand-crafted shrimp
hammer as a bonus gift.</Para>

</Description>

</Product>
```

*Figure 2. A document-centric XML document expressed in XHTML. In the document-centric view, data objects depend on the document as a whole for meaning. The objects' structure is irregular, and data are heterogeneous.*

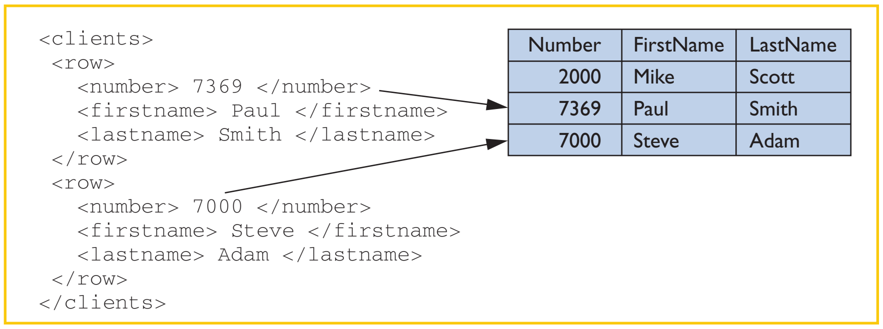
## Major Issues

Since data-centric and document-centric documents serve different purposes, their management requires different approaches. Let's look at the basic issues concerning XML management in object-relational DBMSs.

### Data-Centric XML Document Management

To manage XML data-centric documents, a DBMS must support both data extraction and data formatting services. Data transfer software, either built into the system or available as third-party middleware, often supports data extraction services. Such software can receive XML documents from the network and extract from them structured data to be stored in the DBMS. XML-encoding software, which often supports data formatting services, can take the result of a query, expressed in the DBMS query language, and encode the resulting data in an XML document to be transferred over the network.

To support data extraction, we must define a mapping between XML documents and the DBMS data model. Data extracted from the XML document is generally stored in a set of tables, as shown in Figure 3, and follows a predefined schema. This type of representation is said to be structured. The structured representation doesn't maintain the XML document's original structure. Comments and tag ordering, for example, are lost. In the simplest case — a relational DBMS — the XML document is interpreted according to the following schema:



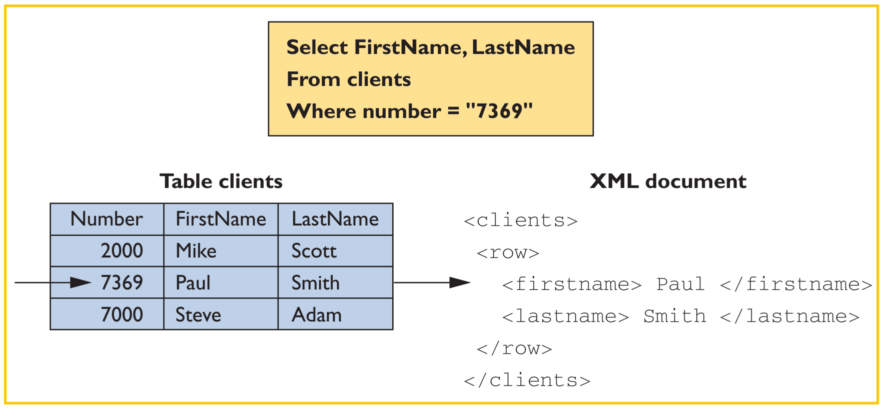
```
<clients>
 <row>
   <number> 7369 </number>
   <firstname> Paul </firstname>
   <lastname> Smith </lastname>
 </row>
 <row>
   <number> 7000 </number>
   <firstname> Steve </firstname>
   <lastname> Adam </lastname>
 </row>
</clients>
```

| Number | FirstName | LastName |
|---|---|---|
| 2000 | Mike | Scott |
| 7369 | Paul | Smith |
| 7000 | Steve | Adam |

*Figure 3. Extracting data from an XML document. After the data is extracted, it is available to queries expressed in the DBMS query language.*

```
<database>
  <table>
    <row>
      <column1>...</column1>
      <column2>...</column2>
      ...
    </row>
      ...
  </table>
    ...
</database>
```

To support data formatting, we must apply a sort of inverse mapping



```
Select FirstName, LastName
From clients
Where number = "7369"
```

| Number | FirstName | LastName |
|---|---|---|
| 2000 | Mike | Scott |
| 7369 | Paul | Smith |
| 7000 | Steve | Adam |

```
<clients>
 <row>
   <firstname> Paul </firstname>
   <lastname> Smith </lastname>
 </row>
</clients>
```

*Figure 4. Data formatting. After a set of tuples is selected from the database with a database query, data formatting services transform the result into XML.*

with respect to data extraction. In particular, after a set of tuples is selected from the database with a database query, data formatting services must transform the result into an XML format. Figure 4 shows such an approach.

### Document-Centric XML Document Management

Dealing with document-centric XML documents requires a DBMS enhanced with new data types for representing XML documents inside databases and new capabilities for querying and managing the documents.

The two types of representation devised for XML storage are *unstructured* and *hybrid* representation. Unstructured representation, shown in Figure 5, characterizes XML documents as single objects – for example, in a single data field. The data type representing such documents varies from one DBMS to another. In general, two approaches are possible: the document can be stored inside the database using, for example, the character large-object (Clob) data type, or it can be stored outside – while remaining linked to – the database. With the first approach, the DBMS is responsible for managing the document; with the second, the local operating system manages the document.

When we don't want to lose the structure of the original document from which data has been extracted, we can combine a structured and an unstructured representation. Moreover, unstructured representation lets us store XML documents with different DTDs inside the same table. When we use the hybrid representation, illustrated in Figure 6, XML documents are stored using part structured and part unstructured representation. Hybrid representation can be useful for XML documents mixing typical structural information (say, a book's structure) with unstructured data (the content of the book's chapters).

To query unstructured XML documents, DBMSs usually extend query languages with specific XML-based selection conditions. Such conditions specify predicates against XML documents and navigate inside XML document structure by using either specific functions made available by the DBMS or standard XML-based languages such as XPath. XML documents stored according to the hybrid representation can be queried by combining selection conditions over the structured and unstructured components.

## XML Support in Commercial DBMSs

Nearly all commercial DBMSs have been extended to handle XML documents. We illustrate XML support in three well-known object-relational DBMSs as it relates to the issues discussed in this column.

### Oracle 8i

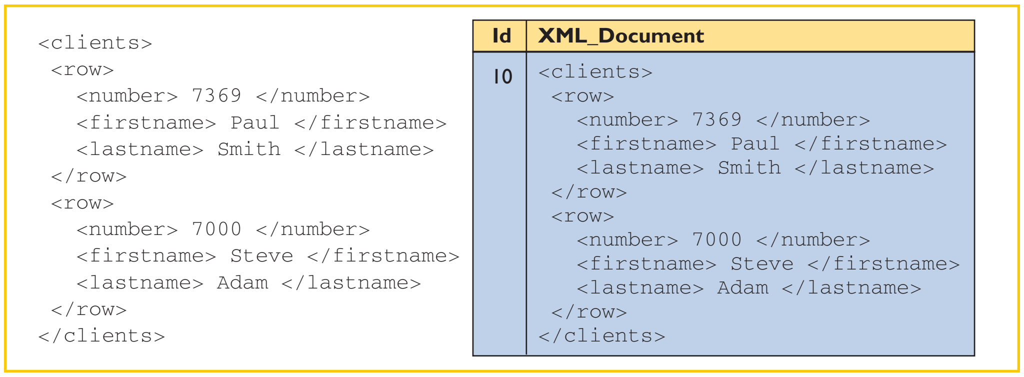
Oracle 8i has an extended architecture with a specific XML layer that supports a set of tools for managing XML documents. In particular, Oracle 8i supports structured, unstructured, and hybrid representation of XML documents. XML-SQL Utility, available as a command-line front-end Java API or a PL/SQL API, supports data extraction and data formatting for data-centric documents. Remember that data extraction requires a specific DTD for XML documents and that data formatting services generate XML documents with a fixed structure. However, users can employ XSL to transform generic XML documents before extraction and after construction.

Document-centric XML documents stored in Oracle 8i use an unstructured representation in CLOB or BFILE fields. With BFILE fields, the XML document is stored and managed outside the database, but metadata for such documents are stored in the DBMS for fast indexing and retrieval. Unstructured and hybrid documents can then be queried using interMedia Context, which provides flexible document indexing for both tag elements and attributes and lets SQL statements query the content of XML documents.

### IBM DB2

In keeping with IBM DB2's traditional architecture, the XML Extender provides DB2 functionalities for storing and managing XML documents. Like Oracle 8i, this architecture handles structured, unstructured, and hybrid documents. Data-centric documents can be stored in an XML collection that represents a set of relational tables containing data extracted from XML documents. The extender supports access and storage methods that let users compose an XML document from existing data, decompose an XML document, and use XML as an interchange data language. DB2 differs from Oracle 8i in that the mapping between XML documents and XML collections is not fixed but can be specified through a data access definition (DAD) document, which specifies the mapping between DTD elements and relational tables and columns by using an XSL Transformations (XSLT) and XPath syntax.

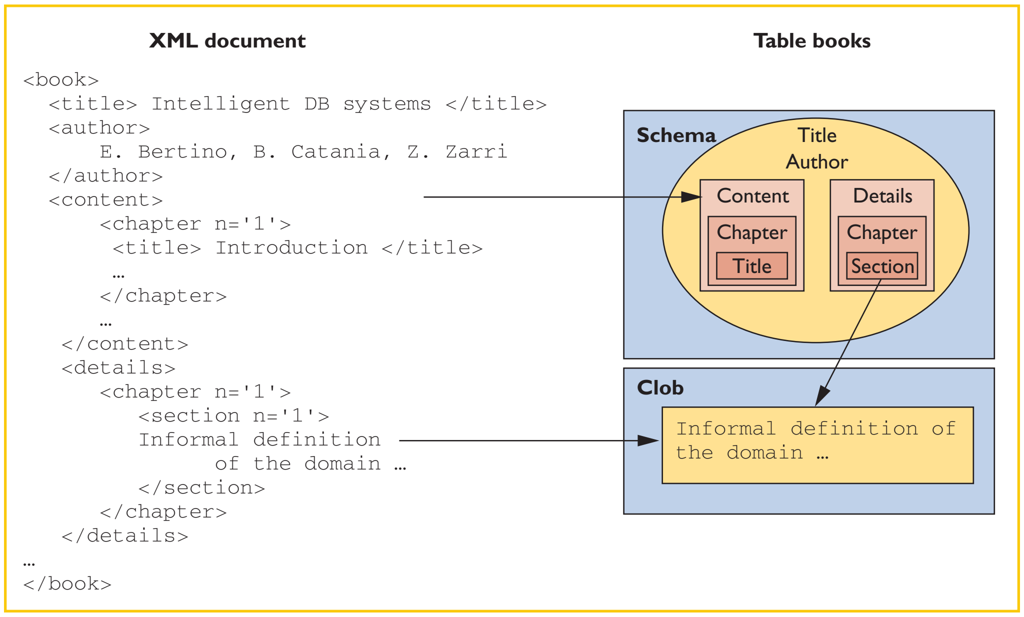
Management of document-centric documents relies on XML columns. DB2 supports three different data types for XML columns: XMLClob, XMLVarChar, and XMLFile, which, respectively, store an XML document as a Clob, a VarChar, or a file on the local file system. In the case of document-centric documents, DAD documents specify which XML elements or attributes should be indexed. User-defined functions (UDFs) support insert, delete, and update operations against XML columns. The IBM Text Extender or specific UDFs enable user queries.



```
<clients>
 <row>
   <number> 7369 </number>
   <firstname> Paul </firstname>
   <lastname> Smith </lastname>
 </row>
 <row>
   <number> 7000 </number>
   <firstname> Steve </firstname>
   <lastname> Adam </lastname>
 </row>
</clients>
```

| Id | XML_Document |
|---|---|
| 10 | `<clients> <row> <number> 7369 </number> <firstname> Paul </firstname> <lastname> Smith </lastname> </row> <row> <number> 7000 </number> <firstname> Steve </firstname> <lastname> Adam </lastname> </row> </clients>` |

*Figure 5. XML storage using unstructured representation. Unstructured representation characterizes an XML document as a single object.*



```
<book>
  <title> Intelligent DB systems </title>
  <author>
      E. Bertino, B. Catania, Z. Zarri
  </author>
  <content>
      <chapter n='1'>
       <title> Introduction </title>
       …
      </chapter>
      …
   </content>
   <details>
      <chapter n='1'>
         <section n='1'>
         Informal definition
               of the domain …
         </section>
      </chapter>
   </details>
…
</book>
```

*Figure 6. A hybrid representation of an XML document. Hybrid representations can be used to characterize XML documents combining structured and unstructured data.*

## Microsoft SQL Server

The Microsoft SQL Server provides several tools for managing data-centric XML documents. In particular, the OpenXML function can extract data from XML documents and store it in relational tables. Extending the Select-From-Where statement with the For XML clause provides XML formatting of a query result. Such a clause specifies to the system that the query output must be formatted as an XML document. SQL Server differs from Oracle 8i in that three different format types can be specified. It also supports an interesting XML view-based approach that permits construction of the so-called XML-data reduced (XDR) schemas. Constructed using an XML-like syntax, such schemas generate views of the database in XML format and can be queried with XPath, either through HTTP or by inserting XPath queries in specific XML templates. A similar approach permits SQL queries. The advantage of such an approach is that the underlying data structure is completely transparent to users, who interact with the DBMS using only XML-based technologies.

## What Next?

Even if most commercial DBMSs currently support XML, several problems remain to be investigated. These mainly concern architectural support for document-centric document management. In particular, limited work has been done on indexing techniques for XML documents.[2] However, developing ad hoc indexing structures for XML documents is crucial for efficient document access. Indexing techniques should address both the content and the structure of XML documents.

Several other issues require attention:

- Persistent XML document storage may benefit from specific clustering algorithms that could help in grouping documents that will be frequently returned together.
- Commercial DBMSs should extend their support to XML query languages, such as XML-QL.
- Development of access control models for XML will guarantee

## XML and Database Resources

**IBM DB2 Universal Database Administration Guide, Version 7, 2001** • http://www-4.ibm.com/cgibin/db2www/data/db2/udb/winos2unix/support/v7pubs.d2w/en_main

**IBM DB2 Universal Database, XML Extender Administration and Programming, Version 7.2, 2001** • http://www-4.ibm.com/software/data/db2/extenders/xmlext/library.html

**IBM DB2 Universal Database, XML Text Extender, 2001** • http://www-4.ibm.com/software/data/db2/extenders/text/index.html

**Microsoft SQL Server** • http://www.microsoft.com/sql/default.asp

**Oracle8i Application Developer's Guide, XML** • http://technet.oracle.com/docs/products/oracle8i/doc_library/817_doc/appdev.817/a86030.pdf; login and password required, registration is free.

**Oracle8i interMedia Text Reference** • http://technet.oracle.com/docs/products/oracle8i/doc_library/17_doc/inter.817/a77063.pdf; login and password required, registration is free.

**Extensible Markup Language (XML) 1.0** • http://www.w3.org/TR/2000/REC-xml-20001006

**XHTML Basic** • http://www.w3.org/TR/xhtml-basic/

**XML Path Language (XPath), Version 1.0** • http://www.w3.org/TR/xpath

**XML-QL** • http://www.w3.org/TR/NOTE-xml-ql/

**Extensible Stylesheet Language (XSL)** • http://www.w3.org/TR/xsl/

**XSL Transformations (XSLT)** • http://www.w3.org/TR/xslt.

secure content-based accesses to XML documents. Some work has already been done on this topic.[3]

Data-centric architectures in particular need truly flexible extraction and formatting mechanisms. Commercial systems typically generate XML documents having specific DTDs. However, researchers have proposed more flexible mechanisms that let users specify the DTD to use for XML document construction.[4] This is particularly important for business-to-business applications, where different parties may need to see the same data organized differently.

Clearly, much work is necessary before we can fully exploit DBMS capabilities in the persistent management of XML documents.

### References

1. R. Bourret, *XML and Databases*, tech. report, Technical Univ. Darmstadt, 2000, http://www.rpbourret.com/xml/XMLAndDatabases.htm.
2. R. Goldman, J. McHugh, and J. Widom, "From Semistructured Data to XML: Migrating the Lorel Data Model and Query Language," *Proc. ACM SIGMOD Workshop the Web and Databases*, INRIA, Paris, 1999.
3. E. Bertino, S. Castano, and E. Ferrari, "Securing XML Documents: The Author-X Project Demonstration," *Proc. ACM SIGMOD Int'l Conf. Management of Data*, ACM Press, New York, 2001.
4. M.F. Fernandez et al., "Publishing Relational Data in XML: The SilkRoute Approach," *IEEE Data Eng. Bull.*, vol. 24, no. 2, 2001, pp. 12-19.

**Elisa Bertino** is a full professor of database systems at the University of Milano, Italy. She is a member of the *IEEE Internet Computing* editorial board.

**Barbara Catania** is a professor of database systems at the University of Genova, Italy.