Pawel DRZYMALA, Henryk WELFLE

Lodz University of Technology

DB2 pureXML - advanced of data storage in the relational - hierarchical structures

Abstract. *The article presents pureXML technology that offers advanced features to store, process and manage XML data in its native hierarchical format. By integrating XML data and relational database structure, users can take full advantage relational data management features. PureXML technology introduced in DB2 to support XML native storage. These are the relational-hierarchical database. Paper presents XQuery and SQL/XML query language that support XML and pureXML data type.*

Streszczenie. *W artykule przedstawiono technologię pureXML, która oferuje zaawansowane funkcje do przechowywania, przetwarzania i zarządzania danymi XML w jego oryginalnym formacie hierarchicznym. Dzięki integracji danych XML i relacyjnej struktury bazy danych, użytkownicy mogą w pełni wykorzystać ich możliwości. PureXML wprowadzono w DB2 w celu natywnego wsparcia przechowywania XML. Jest to relacyjno-hierarchiczna baza danych. Przedstawiono XQuery, SQL/XML, obsługę XML i typ danych pureXML. (**DB2 pureXML - zaawansowanie składowanie danych w strukturach relacyjno-hierarchicznych**).*

Keywords: DB2 pureXML, IBM DB2, relational-hierarchical structures.
Słowa kluczowe: DB2 pureXML, IBM DB2, struktury relacyjno-hierarchiczne.

doi:10.12915/pe.2014.03.36

Introduction

DB2 pureXML offers advanced features to store, process and manage XML data in its native hierarchical format. By integrating XML data and relational database structure, users can take full advantage relational data management features. PureXML, the new technology introduced in DB2 to support XML native storage. These are the relational-hierarchical database. A data model, consisting of nodes of several types linked through ordered parent/child relationships to form a hierarchy.

The pureXML technology in DB2 includes the following capabilities:
- pureXML data type and storage techniques for efficient management XML documents,
- pureXML indexing technology to speed searches of parts XML documents,
- XQuery and SQL/XML query language that support XML,
- support for managing, validating XML Schemes.

Introduction to XML

XML is an acronym for Extensible Markup Language. XML is designed to transport and store data but not to display data. XML tags are not predefined. XML is a W3C recommendation.

XML is an important technology for:
- data integration,
- data exchange,
- web applications and web services,
- service oriented architectures (SOA).

XML documents use a self-describing and simple syntax. Example XML documents (XML document tree Fig.1.):

```
<?xml version="1.0" encoding="utf-8"?>
<!-- comment -->
<customerinfo>
    <name>Kathy Smith</name>
    <addr>
        <country>Canada</country>
        <street>5 Rosewood</street>
        <city>Toronto</city>
    </addr>
    <phone type="work">416-555-1358</phone>
</customerinfo>
```

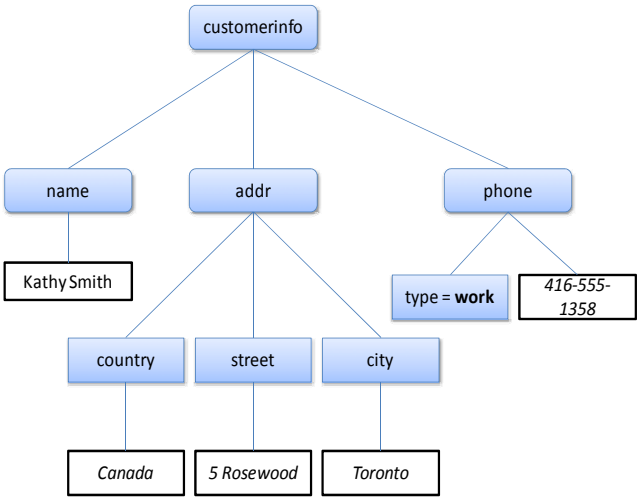


Fig. 1. XML document tree

XML documents must contain a root element. This element is "the parent" of all other elements. The elements in an XML document form a document tree. The tree starts at the root and branches to the lowest level of the tree. In XML all elements must have a closing tag. XML tags are case sensitive. In XML, all elements must be properly nested within each other. XML documents must contain one element that is the parent of all other elements. This element is called the root element. In XML, the attribute values must always be quoted.

Some characters have a special meaning in XML. There are few predefined entity references in XML. The syntax for writing comments in XML is similar to that of HTML. The white-space in a document is not truncated.

An element XML can contain:
- other elements,
- text,
- attributes,
- a mix of all of the above.

In the example above, <customerinfo> and <addr> have element contents, because they contain other elements. <addr> also has an attribute (country="Canada"). <name>, <street> and <city>have text content because they contain text.

Attributes provide additional information about an element and must be quoted. There are no rules about when to use attributes or when to use elements. Elements are more useful in XML.

Example XML documents with attributes:

```
<addr country="Canada">
    <street>5 Rosewood</street>
    <city>Toronto</city>
</addr>
```

Example XML documents with elements:

```
<addr>
    <country>Canada"</country>
    <street>5 Rosewood</street>
    <city>Toronto</city>
</addr>
```

In the first example country is an attribute. In the last, country is an element. Both examples provide the same information.

Problems with using attributes:
- cannot contain multiple values,
- cannot contain tree structures,
- are not easily expandable for future changes.

Attributes are difficult to read and maintain. Use elements for data. Use attributes for information not relevant.

A "Well Formed" XML document has correct XML syntax.

The XML syntax rules:
- documents must have a root element,
- elements must have a closing tag,
- tags are case sensitive,
- elements must be properly nested,
- attribute values must be quoted.

A "Valid" XML document is a "Well Formed" XML document (Fig.2), which also conforms to the rules of a Document Type Definition (DTD) or XML Schema.

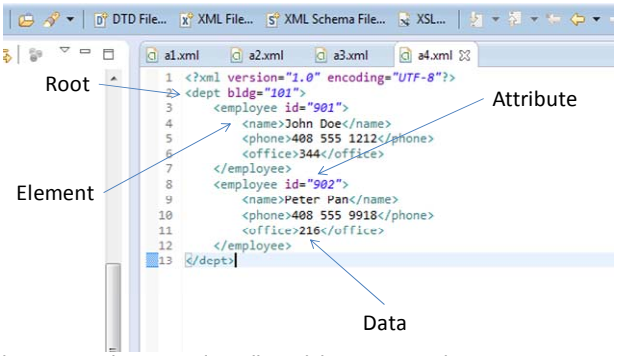


Fig. 2. XML document in Eclipse (elements XML)

DB2 pureXML technologies

XML documents can be stored in text files, XML repositories or databases. XML data are stored in databases because it provide:
- managing large volumes of XML data is a database problem,
- integration new XML data with existing relational data. Moreover, relational data can be published as XML, and vice versa.

There are two types of databases for storing XML data:
- XML-enabled databases,
- Native XML databases.

An XML-enabled database uses the relational model as its core data storage model to store XML. This requires either a mapping between the XML (hierarchical) data model and the relational data model, or else storing the XML data as a character large object. While this can be considered as "old" technology, it is still being used by many database vendors.

The storage format is the same as the processing format: there is no mapping to the relational model, and XML documents are not stored as unparsed strings (CLOBs or varchars). When XPath or XQuery statements are used, they are processed natively by the engine, and not converted to SQL. DB2 is currently the commercial data server providing this support.

In DB2, there are now four ways to access data (Fig.3):
- use SQL to access relational data,
- use SQL with XML extensions (SQL/XML) to access XML data,
- use XQuery to access XML data,
- use XQuery to access relational data.

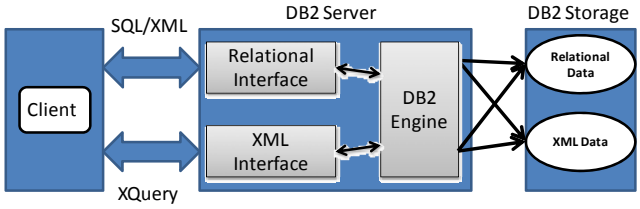


Fig. 3. Access to relational and XML data in DB2

The pureXML technology in DB2 includes the following capabilities:
- pureXML data type and storage techniques for efficient management XML documents,
- pureXML indexing technology to speed searches of parts XML documents,
- XQuery and SQL/XML query language support XML,
- support for managing, validating XML Schemes,
- comprehensive administrative capabilities,
- integration with popular APIs and development environments.

PureXML technology advantages:
- documents are stored in columns of tables using the new XML data type (Fig.4, 5, 6). Relational data and hierarchical data (XML documents) are both stored in a DB2 database.

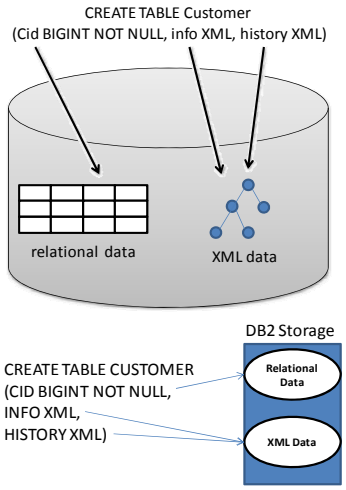


Fig. 4. New XML type in DB2

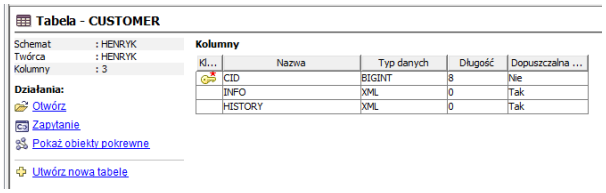


Fig. 5. Table Customer with XML columns

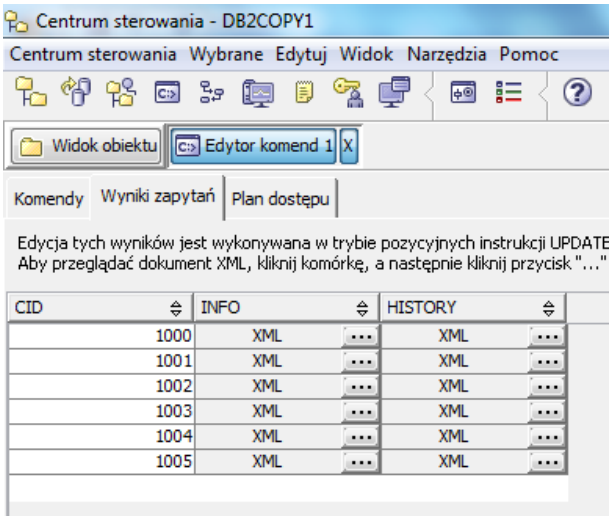
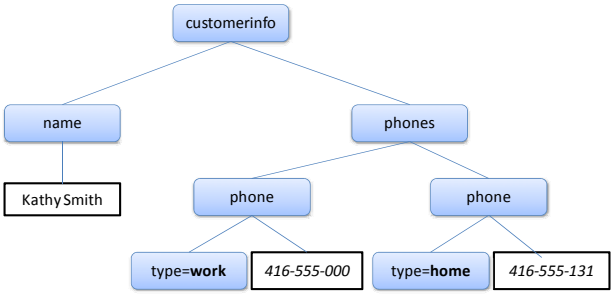


Fig. 6. Data in Customer table with XML columns.

- XML can reduce code complexity,
- changes to schema are easier using XML and pureXML technology. If there is a table with a phone number and is needed to add the next phone numbers it is necessary to change the structure of a relational database. Using XML technology just add a new number in the XML structure (Fig. 7).




```
<customerinfo>
        <name>Kathy Smith</name>
        <phones>
                <phone type='work'>416-555-000</phone>
                <phone type='home' >416-555-131</phone>
        <phones>
</customerinfo>
```

Fig. 7. Changes to schema with use XML

- XML can improve application performance.

Optimization and performance

Creating a database for storing XML data in DB2 is no different from creating a database for relational data. Both types of data can be stored and accessed simultaneously in the same database. XML data is typically represented as Unicode, the database can be created using a Unicode setting. We can create a table with both relational and XML data types (Fig.4).

To improve performance type XML, you should use indexes. In an XML document, indexes can be created for elements, attributes or for values (text nodes) (Fig. 7).

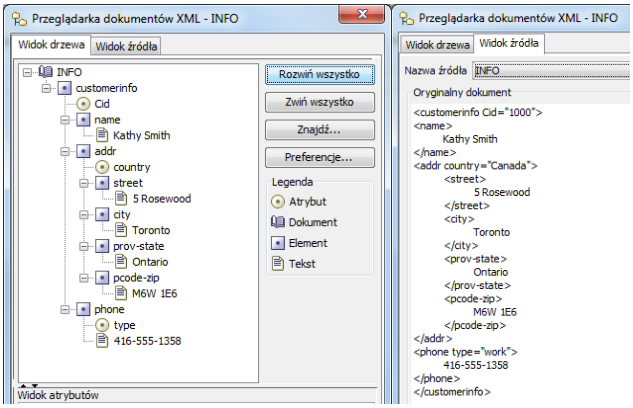


Fig. 8. Elements, attributes and values in an XML document.

Creates an index on the attribute @type – example statement (table customer, column info XML type, structure as fig. 4, 5):

```
CREATE UNIQUE INDEX index1 ON customer(info)
GENERATE KEY USING
xmlpattern '/customerinfo/phones/phone/@type'
AS sql VARCHAR(10)
```

Creates an index on the element name or all element – example statement:

```
CREATE INDEX index1 ON customer(info)
GENERATE KEY USING
xmlpattern '/customerinfo/name'
AS sql VARCHAR(10)
```

Creates an index on all text nodes (all values) (not recommended - the index would be too large):

```
CREATE INDEX index1 ON customer(info)
GENERATE KEY USING
xmlpattern '//text()'
AS sql VARCHAR(10);
```

XML data can be easily entered into a table created with XML data types using the INSERT statement (deleted or updated) - example statement:

```
INSERT INTO customer (Cid, info) VALUES (1,
'<customerinfo>
<name>Kathy Smith</name>
        <phones>
        <phone type='work'>416-555-000</phone>
        <phones>
</customerinfo> ');
```

Querying XML data with SQL/XML

Using regular SQL statements it allows to work with rows and columns. An SQL statement can be used to work with full XML documents, however, it would not help when attempting to retrieve only part of the document. In such cases, it is needed to use SQL with XML extensions (SQL/XML). It can be used to present XML data in relational form. SQL/XML provides many features. We have the following features xmlparse, xmlserialize, xmlvalidate, xmlexists, xmlquery, xmltable, xmlcast and other.

An example using XMLEXISTS:

```
SELECT * FROM customer
WHERE XMLEXISTS('$R/customerinfo/@Cid="1000" '
PASSING info AS "R");
```

An example using XMLQUERY:

```
SELECT
xmlquery('$doc/customerinfo/name'
passing info as "doc") as empinfo
FROM CUSTOMER;
```

IBM Data Studio allows to help with writing queries using the SQL / XML (Fig. 9).

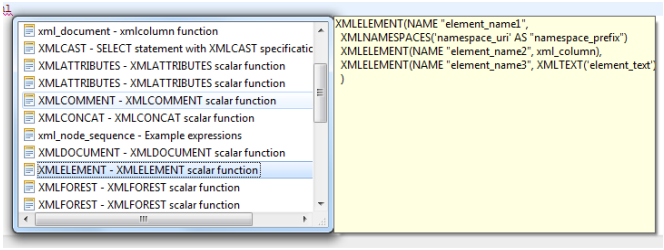


Fig. 9. IBM Data Studio – context help.

Querying XML data with XQuery

XQuery is a query language created for XML. XQuery supports XPath (subset of XQuery) to navigate the XML structure. XQuery is case sensitive and returns sequences of XML data. XQuery supports the FLWOR expression.

FLWOR is an acronym:

- FOR: iterates through a sequence, binds a variable to items,
- LET: binds a variable,
- WHERE: eliminates items,
- ORDER: reorders items,
- RETURN: query results.

It is an expression that allows manipulation of XML documents, enabling to return another expression.

IBM Data Studio allows to help with writing queries using the XQuery (Fig. 10).

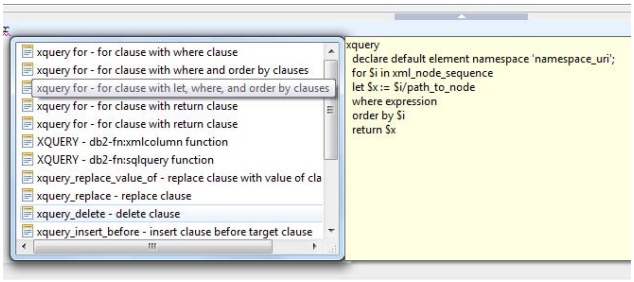


Fig. 10. IBM Data Studio – context help.

Examples using XQuery:

The query returns list all the customer names in alphabetical order. Select only the customers who live in the city of Lodz. Return the customer name and address information, without creating new elements. Use INFO column in CUSTOMER table.

```
xquery
for $customer in db2-fn:xmlcolumn
('XMLCUSTOMER.INFO')/customerinfo
where $customer/addr/city = 'Lodz'
order by $customer/name
return ($customer/name, $customer/addr);
```

The query returns all the XML documents from INFO column in CUSTOMER table.

```
xquery
db2-fn:sqlquery('select info from customer');
```

or

```
xquery
db2-fn:xmlcolumn('CUSTOMER.INFO');
```

The query returns all the names of customers where the addr, city is “Lodz”.

```
xquery
db2-fn:xmlcolumn
('CUSTOMER.INFO')/customerinfo[addr/city='Lodz']/
name;
```

The query returns list the country and city for each customer, in a new element XML called custAddr.

```
xquery
for $customer in db2-fn:xmlcolumn
('CUSTOMER.INFO')/customerinfo
return
<custAddr> {$customer/addr/@country,
$customer/addr/city} </custAddr>;
```

The query returns count the number of products whose price is less than 100. Function count is built-in functions. In XML, a name can be qualified by a namespace.

```
xquery
count(db2-fn:xmlcolumn
('XMLPRODUCT.DESCRIPTION')/product/description[pri
ce < 100]);
```

Conclusion

Article presents XML and pureXML technology. PureXML offers advanced features to store, process and manage XML data in its native hierarchical format. By storing XML documents in a DB2 database can be used advantage of security, performance and coding flexibility using pureXML. PureXML is a technology that allows to store the XML documents in hierarchical format (format a tree). The tree for the XML document was already built and stored in the database. In addition, pureXML technology uses a native XML engine. It presents query XML documents using SQL/XML and XQuery.

REFERENCES

[1] Smarter Planet IBM: http://www.ibm.com/smarterplanet/us/en/overview/ideas/
[2] Raul F. Chong, Clara Liu, Sylvia F. Qi, Dwaine R. Snow: Understanding DB2: Learning Visually with Examples
[3] IBM DB2 10.1 dla systemów Linux, UNIX i Windows - Centrum informacyjne: http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/index.jsp
[4] IBM DB2 10.5 dla systemów Linux, UNIX i Windows - Centrum informacyjne: http://publib.boulder.ibm.com/infocenter/db2luw/v10r5/index.jsp
[5] *Matthias Nicola, Pav Kumar-Chatterjee:* DB2 pureXML Cookbook: Master the Power of the IBM Hybrid Data Server, IBM Press book, 10 Aug 2009.
[6] Whei-Jen Chen, A. Sammartino, D. Goutev, F. Hendricks, I.Komi, Ming-Pang Wei, R. Ahuja: DB2 9 pureXML Guide, ibm.com/redbooks, January 2007.
[7] C.M. Saracco, D. Chamberlin, R. Ahuja: DB2 9: pureXML Overview and Fast Start, ibm.com/redbooks, June 2006.

***Authors**: Ph.D. Pawel Drzymala, Lodz University of Technology, Institute of Mechatronics and Information Systems, Stefanowskiego 18/22, 90-924 Lodz, E-mail: pawel.drzymala@p.lodz.pl; Ph.D. Henryk Welfle, Lodz University of Technology, Institute of Mechatronics and Information Systems, Stefanowskiego 18/22, 90-924 Lodz, E-mail: henryk.welfle@p.lodz.pl.*